CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE CONTROL AUTOMÁTICO

Análisis y Diseño de un Robot Humanoide: De Estructura Estática a Robot Caminante

Tesis que presenta

Ing. Andrés Enríquez Cobo

Para obtener el grado de

Maestro en Ciencias

En la Especialidad de

Control Automático

Director de Tesis:

Dr. Juan Manuel Ibarra Zannatha

México, D.F. Febrero 2015

Resumen

Este trabajo de Tesis presenta la metodología básica que debe seguirse para trabajar con un robot humanoide, ésta abarca desde su modelado cinemático hasta su caminado y percepción del entorno.

Los modelos cinemáticos desarrollados se validaron por medio de interfaces gráficas, utilizando un modelo virtual del humanoide. El modelo de la pierna se obtuvo mediante la metodología de Denavit-Hartenberg, al igual que el modelo directo del brazo. Sin, embargo, el modelo inverso del brazo, se obtuvo mediante análisis geométrico. Además, se obtuvo un modelo dinámico simplificado, el cual se validó utilizando la caja de herramientas de SimMechanics de Matlab.

El humanoide con el cual se trabajó, fue desarrollado en su totalidad en el Departamento de Control Automático del CINVESTAV en conjunto con estudiantes de la Universidad La Salle como parte de la línea de investigación sobre robots humanoides. El diseño de este humanoide fue modificado para incorporar dos controladores: uno se utiliza para el cálculo de trayectorias articulares y la toma de decisiones por medio de algoritmos de visión; el otro se utiliza para interpretar estas decisiones y ejecutarlas mediante los servomotores. Este sistema de comunicación tipo cerebro-cerebelo, a pesar de estar implementado en otros prototipos como el robot DARwIn, utiliza componentes cuya comunicación no había sido establecida, proporcionando así, una extensión al uso de ambos controladores.

El nuevo prototipo cuenta con una cámara estereoscópica. Ésta permite crear aplicaciones más complejas de visión artificial por medio del cálculo de distancia en el mundo tridimensional, lo cual proporciona una ventaja sobre los prototipos comerciales existentes ya que, por lo general, cuentan con únicamente una cámara. Esta cámara además, nos proporciona el lazo de control necesario para manipular objetos por medio de los brazos y permite realizar el seguimiento de objetos al integrar el modelado y la percepción.

Con respecto problema de caminado, se presentan dos algoritmos: un caminado parametrizado y un caminado omnidireccional. Ambos algoritmos se validaron en base al criterio del ZMP.

Finalmente, se presentan los resultados, tanto en las interfaces desarrolladas como en el prototipo físico, de los algoritmos desarrollados en ésta Tesis.

Abstract

This thesis presents the basic methodology that should be followed to work with a humanoid robot, this methodology covers from kinematic modeling till walking and environment perception.

The kinematic models developed were validated through graphical user interfaces, using a virtual model of the humanoid. The leg's model was obtained through Denavit-Hartenberg's methodology, as well as the direct model of the arms. However, the inverse model of the arm, was obtained through geometric analysis. In addition, a simplified dynamic model was obtained, which was validated using Matlab's SimMechanics Toolbox.

The humanoid used to work with, was developed completely at the Departamento de Control Automático of CINVESTAV in a group with students from the Universidad La Salle as a part of the research line about humanoid robots. This humanoid's design was modified to include two controllers: one is used to compute the articular trajectories and decision making through vision algorithms; the other one is used to interpret these decisions and executed through the servomotors. This communication system type cerebrum-cerebellum type, despite being implemented in other prototypes like the DARwIn robot, uses components which had no communication established between them, providing an extension to the usage of both controllers.

The new prototype has a stereo camera. This allows to create more complex artificial vision applications by computing distance in the three-dimensional world, which provides an advantge over existing commercial prototypes due to the fact that, in general, they have only one camera. Besides, this camera provides the necesarry control loop to manipulate objects using the arms and allows us to perform objects tracking by combining the modeling and perception.

Regarding the walking problem, two algorithms are presented: a parameterized walking and an omnidirectional walking. Both algorithms were validated using the ZMP criterion.

Finally, results are presented, as much in the developed interfaces as in the physical prototype, of the algorithms developed in this Thesis.

Agradecimientos

- A CONACYT, por haberme otorgado la beca que me permitió dedicarme de lleno a mis estudios de Maestría.
- A mi familia, por apoyarme en todo momento.
- A Gustavo Alemán, por su ayuda y por enseñarme que es posible trabajar en equipo.
- A Hazel, Yasmyn e Ionela, por su invaluable apoyo y amistad.
- Al Ing. Roberto Lagunes Feregrino, por sus recomendaciones, su tiempo y su ayuda que permitieron que este trabajo saliera adelante.
- A mis compañeros de la Maestría: Ricardo Carrillo Mendoza, Emma Mireya Sánchez Mosqueda, Christian Abraham Enríquez Olguin, Iván Trejo Zúñiga, Gian Carlo Gómez Cortes y Christopher Diego Cruz Ancona. Su apoyo y sus pláticas significaron más de lo que creen.
- Al Dr. Cisneros y al Dr. Moragues, mis mejores profesores, por enseñarme que la mente creativa e independiente puede llegar lejos.
- A Rafael Stanley Núñez Cruz y a Diego Alberto Bravo Montenegro, Doctores sin grado aún, por su tiempo y sus valiosas aportaciones al presente trabajo.
- Al Dr. Ahmed Chemori, del Laboratorio de Informática, de Robótica y de Microelectrónica de Montpellier (LIRMM) en Montpellier, Francia, quien me aceptó para realizar una estancia de investigación.
- A mis sinodales: el Dr. Juan Manuel Ibarra Zannatha, el Dr. Alejandro Justo Malo Tamayo y el Dr. Alejandro Aceves López, por el tiempo invertido en revisar el trabajo de Tesis.

Índice general

1. Introducción **1**
- 1.1. Estado del arte . . . 3
- 1.2. Planteamiento del problema . . . 6
- 1.3. Propósito del trabajo de Tesis . . . 7
- 1.4. Estructura del trabajo de Tesis . . . 8

2. Descripción del Robot Humanoide AH1N2 **11**
- 2.1. Sistema Mecánico . . . 12
  - 2.1.1. Modificaciones del nuevo diseño . . . 12
    - 2.1.1.1. Tórax . . . 12
    - 2.1.1.2. Brazos . . . 14
    - 2.1.1.3. Cabeza . . . 15
  - 2.1.2. Servomotores . . . 15
- 2.2. Sistema Computacional . . . 16
  - 2.2.1. Roboard RB-100 . . . 16
  - 2.2.2. CM-700 . . . 17
  - 2.2.3. RoboPlus . . . 18
  - 2.2.4. Minoru 3D Webcam . . . 18
  - 2.2.5. Comunicación . . . 19
    - 2.2.5.1. Roboard-CM . . . 19
    - 2.2.5.2. Roboard-Minoru . . . 20
    - 2.2.5.3. CM700-Servomotores . . . 21
- 2.3. Sistema Electrónico . . . 21
  - 2.3.1. Baterías . . . 22

3. Modelado **23**
- 3.1. Modelo Cinemático . . . 23
  - 3.1.1. Metodología de Denavit-Hartenberg . . . 23
  - 3.1.2. Cinemática de las piernas . . . 25
    - 3.1.2.1. Cinemática Directa . . . 25
    - 3.1.2.2. Cinemática Inversa . . . 27
  - 3.1.3. Cinemática de los Brazos . . . 30
    - 3.1.3.1. Cinemática Directa . . . 30
    - 3.1.3.2. Cinemática Inversa . . . 32
- 3.2. Modelado Virtual . . . 37
  - 3.2.1. Unidad Locomotora . . . 38
    - 3.2.1.1. Simulador Cinemático . . . 38
  - 3.2.2. Unidad Pasajera . . . 39

3.2.2.1. Simulador Cinemático 40
3.2.3. Modelo Completo 40
3.3. Modelo Dinámico 41
3.3.1. Formulación Newton-Euler 41
3.3.2. SimMechanics 44

4. Caminado Bípedo **47**
4.1. Conceptos 47
4.1.1. Fases del caminado 47
4.1.2. Planos de movimiento 48
4.2. Caminado Parametrizado 49
4.3. Caminado Omnidireccional 51
4.3.1. Interfaz de la pierna 52
4.3.2. Reloj Central 54
4.3.3. Vectores de entrada 54
4.3.4. Desplazamiento 54
4.3.5. Acortamiento 55
4.3.6. Carga 56
4.3.7. Balanceo 56
4.3.8. Equilibrio 57
4.3.9. Resultado 57
4.4. Estabilidad del caminado 58
4.4.1. Derivación del Zero-Moment Point 58
4.4.2. Cálculo del ZMP 60
4.4.3. Modelo simplificado 61

5. Sistema de Visión **63**
5.1. Visión Estereoscópica 63
5.1.1. Calibración de cámaras 64
5.1.2. Cálculo de distancia 3D 65
5.2. OpenCV 66
5.2.1. Calibración de Cámaras 66
5.2.2. Distancia 3D 68
5.3. Coordinación Percepción-Control 70
5.3.1. Cinemática de la cámara 70
5.3.2. Detección de objetos 72
5.3.3. Seguimiento y alcance de objetos 73

6. Pruebas y Resultados **75**
6.1. Cinemática 75
6.1.1. Piernas 75
6.1.2. Brazos 79
6.2. Dinámica 84
6.3. Caminado 87
6.3.1. Caminado parametrizado 87

6.3.2. Caminado omnidireccional 87
6.3.3. Estabilidad del caminado 90
6.4. Coordinación Percepción-Control 91
6.4.1. Segmentación 91
6.4.2. Seguimiento 92
6.4.3. Alcance de objeto 92

7. Conclusiones **93**
7.1. Trabajo a futuro 96

A. Especificaciones del Hardware **99**

B. Configuración de la Roboard **103**
B.1. Instalador de Lubuntu 103
B.2. Imagen de Lubuntu 104

C. Instalación de OpenCV **107**

D. Conversión de Datos: double-unsigned char-double **109**

E. Código VRML **111**
E.1. Unidad Pasajera 111
E.2. Unidad Locomotora 115

F. Código Matlab **123**
F.1. Cinemática Piernas 123
F.2. Cinemática Brazos 124
F.3. Función del Modelo Dinámico 126

G. Código C **131**
G.1. Función de Caminado Parametrizado 131
G.2. Función de Caminado Omnidireccional 133
G.3. Seguimiento y alcance de objetos 135

Índice de figuras

1.1. Primer Robot Industrial creado por la compañía Unimation en 1954 [2] 2
1.2. WABOT-1 [2] 4
1.3. Robots desarrollados por Honda 4
1.4. Robot Humanoide Roboray de Samsung [9] 6

2.1. Conexón serie de 2 cadenas 12
2.2. Tórax del prototipo anterior 13
2.3. Tórax del nuevo prototipo 13
2.4. Comparación de brazos de ambos prototipos 14
2.5. Comparación de cabezas de ambos prototipos 15
2.6. Roboard RB-100 16
2.7. Controlador CM-700 17
2.8. Conexiones de los servomotores al controlador CM-700 17
2.9. Minoru 3D Webcam 18
2.10. Conversión de datos para transmisión 20
2.11. Arreglo de caracteres para transmisión 20

3.1. Diagrama Cinemático de las Piernas 26
3.2. Diagrama cinemático de los brazos 31
3.3. Triángulo formado por el brazo y el vector de posición 33
3.4. Triángulo con flexión-extensión del codo y SH 33
3.5. Proyección en el plano sagital 34
3.6. Cálculo de θ_2 36
3.7. Cálculo de θ_3 36
3.8. Modelo VRML de la Unidad Locomotora 37
3.9. Interfaz de la unidad locomotora 38
3.10. Modelo VRML de la Unidad Pasajera 39
3.11. Puntos de vista de las cámaras 39
3.12. Interfaz de la unidad pasajera 40
3.13. Modelo VRML completo 41
3.14. Propiedades del eslabón Cadera 45
3.15. Diagrama de bloques de SimMechanics 45
3.16. Subsistema de cada articulación 46
3.17. Comparación de modelos ensamblados 46

4.1. Planos de movimiento [21] 48
4.2. Parámetros del caminado en el plano sagital [22] 49
4.3. Parámetros del caminado en el plano frontal [22] 50
4.4. Ángulos de Roll, Pitch y Yaw de la pierna y del pie 52

4.5. Análisis geométrico para obtener el ángulo de la rodilla 53
4.6. Modelo de péndulo para la pierna 54
4.7. Fuerzas que actúan sobre el pie del robot. [24] 59
4.8. Modelo dinámico simplificado para el cálculo del ZMP. [2] 61

5.1. Sistema estereoscópico [26] 65
5.2. Tablero para calibración [27] 66
5.3. Configuración de la escena para medición 3D 68
5.4. Imágenes segmentadas 69
5.5. Distancia calculada a partir de dos imágenes 70
5.6. Diagrama cinemático de la cabeza 71

6.1. Cinemática directa de la pierna en la prueba uno 76
6.2. Cinemática inversa de la pierna en la prueba uno 77
6.3. Poses obtenidas mediante cinemática inversa de la pierna en la prueba uno 77
6.4. Cinemática directa de la pierna en la prueba dos 78
6.5. Cinemática inversa de la pierna en la prueba dos 78
6.6. Poses obtenidas mediante cinemática inversa de la pierna en la prueba dos 79
6.7. Cinemática inversa del brazo en la prueba uno 80
6.8. Poses obtenidas mediante cinemática inversa del brazo en la prueba uno 80
6.9. Cinemática inversa del brazo en la prueba dos 81
6.10. Poses obtenidas mediante cinemática inversa del brazo en la prueba dos 81
6.11. Cinemática inversa del brazo en la prueba tres 82
6.12. Poses obtenidas mediante cinemática inversa del brazo en la prueba tres 82
6.13. Cinemática inversa del brazo en la prueba cuatro 83
6.14. Poses obtenidas mediante cinemática inversa del brazo en la prueba cuatro 83
6.15. Pares de la pierna de soporte 85
6.16. Pares de la pierna flotante 86
6.17. Caminado parametrizado en el plano sagital 88
6.18. Caminado parametrizado en el plano frontal 88
6.19. Caminado omnidireccional en el plano sagital 89
6.20. Caminado omnidireccional en el plano frontal 89
6.21. Trayectoria del ZMP en el caminado parametrizado 90
6.22. Trayectoria del ZMP en el caminado omnidireccional 90
6.23. Trayectoria del ZMP respecto al polígono de soporte simple 90
6.24. Segmentación de pelota 91
6.25. Detección de pelota 91
6.26. Seguimiento de pelota roja utilizando servomotores de Pan y *Tilt* . . . 92
6.27. Alcance de pelota con brazo izquierdo 92

A.1. Especificaciones del subcontrolador CM-700 99
A.2. Especificaciones de la Roboard RB-100 100
A.3. Especificaciones del servomotor RX-28 101
A.4. Especificaciones del servomotor RX-64 102

Índice de tablas

2.1. Principales propiedades de los servomotores 15
2.2. Consumo energético en modo de espera 22

3.1. Tabla de parámetros de DH 24
3.2. Tabla de parámetros de la pierna 25
3.3. Tabla de parámetros del brazo 31

5.1. Tabla de parámetros de la cabeza 71

6.1. Tabla de comparación de la cinemática inversa del brazo en la prueba uno 84
6.2. Tabla de comparación de la cinemática inversa del brazo en la prueba dos 84
6.3. Tabla de comparación de la cinemática inversa del brazo en la prueba tres 84
6.4. Tabla de comparación de la cinemática inversa del brazo en la prueba cuatro 84

Capítulo 1

Introducción

Desde el nacimiento, las personas han tenido que trabajar para mantener su forma de vida, ya sea para asegurar su supervivencia o simplemente para tratar de vivir más cómodamente. Esto es algo que no cambia de persona a persona, cada quien debe trabajar para asegurar su propia existencia. Es debido a esto que las personas siempre han soñado con ser capaces de fabricar algo que realice el trabajo por ellas. En este contexto es donde surge el término de Robot.

Definición 1.1 (Robot). Un *robot* es un mecanismo actuado programable en dos o más ejes con un grado de autonomía, moviéndose dentro de su entorno, para realizar tareas deseadas. Autonomía en este contexto significa la habilidad de realizar tareas deseadas basadas en el estado actual y medición, sin intervención humana. [1]

Desde la Revolución Industrial se han construido máquinas que pueden agilizar el trabajo realizado por una persona pero esto no ha llegado al grado de reemplazarlas completamente. Las máquinas dependen de alguien que las opere y supervise sus actividades. Además, las máquinas diseñadas se construyeron para realizar una tarea específica. Esto quiere decir que la máquina podía realizar eficientemente la actividad para la cual fue construida pero esto perdía importancia en el momento que se deseaba utilizar la misma máquina para realizar otra actividad, esto significaba que la máquina original debía pasar por un cambio drástico o incluso debía ser reemplazada por completo. El deseo de tener máquinas flexibles que pudieran ser utilizadas en diversas actividades fue lo que generó la Robótica del Siglo XX, la Robótica Industrial.

Definición 1.2 (Robot Industrial). Un *robot industrial* es un manipulador multipropósito, reprogramable, automáticamente controlado, programable en tres o más ejes, que puede ser fijo o móvil para uso en aplicaciones industriales de automatización. [1]

La Robótica Industrial comenzó con el desarrollo de los robots manipuladores mostrados en la Figura 1.1, capaces de realizar múltiples actividades dentro de una línea de ensamble sustituyendo así el trabajo manual y repetitivo de los trabajadores reduciendo costos de producción.

Figura 1.1: Primer Robot Industrial creado por la compañía Unimation en 1954 [2]

Estos robots se utilizan dentro de un área de trabajo delimitada y acondicionada para cada robot. Además de realizar tareas repetitivas, son capaces de realizarlas a gran velocidad con gran precisión.

Este tipo de robots a pesar de haber revolucionado la industria, presentan limitaciones para alcanzar el objetivo original de crear una máquina autónoma que pueda realizar trabajo humano (no sólo en la industria). Algunas de estas limitaciones son:

- El robot se encuentra fijo dentro de su espacio de trabajo.
- Necesidad de un ambiente de trabajo acondicionado para el robot.
- Programados para realizar una tarea a la vez.

Con el fin de superar estas limitaciones es como surge la Robótica del Siglo XXI, la Robótica de Servicio.

Definición 1.3 (Robot de Servicio). Un *robot de servicio* es un robot que desempeña tareas útiles para humanos o equipo excluyendo su aplicación en automatización industrial. [1]

Es aquí donde aparecen los robots móviles cuyo espacio de trabajo está limitado por la compatibilidad entre su medio de locomoción y el entorno donde se encuentran.

Los robots móviles se clasifican de acuerdo al medio donde se desplazan, como: robots terrestres, aéreos, acuáticos, etc. Dentro de los robots terrestres se encuentran principalmente los robots que utilizan las ruedas como medio de locomoción y los que utilizan piernas. Es aquí donde se encuentran los robots bípedos humanoides, sobre los cuales se enfocará el trabajo de tesis.

1.1. Estado del arte

A pesar de que la Robótica de Servicio es del Siglo XXI, los intentos por reproducir sistemas mecánicos que imiten a un humano son más antiguos. Por ejemplo en el Siglo XII Al-Jazari diseñó un autómata humanoide, en el Siglo XV Leonardo Da Vinci diseñó otro autómata humanoide y en Japón existe una tradición de crear muñecas mecánicas llamadas karakuri ningyo, que existe por lo menos desde el Siglo XVIII. Posteriormente en el Siglo XX se introdujo la animatrónica en los parques de atracciones.

A mediados del Siglo XX muchos investigadores desarrollaron sistemas para medición, locomoción y manipulación, inspirados por habilidades humanas. Estos sistemas desarrollados se encontraban de manera aislada y en 1973 la Universidad de Waseda en Japón los integró en un robot humanoide: WABOT-1 mostrado en la Figura 1.2

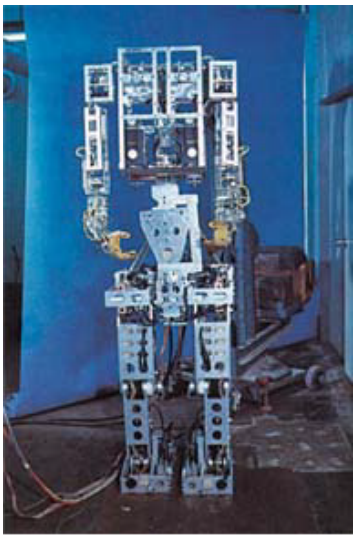


Figura 1.2: WABOT-1 [2]

Esta serie de robots integró funciones que han estado bajo constante investigación desde entonces como: reconocimiento visual de objetos, generación y reconocimiento del habla, manipulación de objetos con ambas extremidades, y caminado bípedo.

En 1996, Honda reveló el Humanoide P2 mostrado en la Figura 1.3a, el resultado de un proyecto confidencial que empezó en 1986.

(a) Honda P2 (180 cm de altura, 210 kg) [2]

(b) Honda Asimo (120 cm de altura, 43 kg) [2]

Figura 1.3: Robots desarrollados por Honda

El robot humanoide P2 fue el primer robot humanoide de tamaño natural capaz de caminar establemente de manera bípeda con fuente de energía y procesamiento abordo. Esta serie de robots humanoides presentaron un gran adelanto comparados con los robots desarrollados en instituciones académicas debido a que utilizaron actuadores armónicos de alto par, así como piezas ligeras de alta rigidez específicas.

El proyecto de humanoides de Honda actualmente se encuentra en desarrollo con su modelo ASIMO (Advanced Step in Innovative MObility) mostrado en la Figura 1.3b. Éste es capaz de correr a 9km/h, saltar en un pie o ambos, caminar en superficies irregulares, subir y bajar escaleras, interactuar con personas, manipulación con ambas manos, etc.

Al mismo tiempo en Estados Unidos se desarrolló el proyecto Cog en 1993, proyecto enfocado a crear un humanoide que aprendiera a pensar y posteriormente elevar su nivel de abstracción para realizar nuevas tareas. Este proyecto, al centrarse en las ciencias cognitivas y biológicas, consistió únicamente en el diseño de la parte superior del cuerpo. [2]

En 2006 se creó el grupo AAUBOT en la Universidad de Aalborg en Dinamarca. Este grupo tiene como propósito construir e implementar un robot humanoide operacional del tamaño de un humano de que cierre la brecha entre salud, tecnología y robótica. Algunos de los robots diseñados hasta la fecha son AAUrobot I [3], AAU-BOT1 [4] y Roberto [5]. Posteriormente en la competencia de estudiantes de diseño de mecanismos de la ASME de ese año un grupo de estudiantes de Virginia Tech, asesorados por Dennis Hong, ganó el segundo lugar al presentar el diseño del robot humanoide DARwIn (Dynamic Anthropomorfic Robot with Intelligence) [6].

En 2011 en Japón, Masahiko Yamaguchi (Dr. Guero) modificó un robot Kondo KHR-3HV y lo equipo con un giroscopio y un acelerómetro que transmiten información a un controlador que hace ligeros ajustes a la posición de sus brazos y piernas para mantener su equilibrio. Utilizando este robot ha logrado una marcha que logra imitar de manera casi idéntica la marcha de un humano, además de que ha puesto a prueba sus algoritmos en condiciones de equilibrio críticas como al caminar sobre una cuerda floja. [7] [8]

Recientemente, en 2013, el Laboratorio de Robótica de Bristol en conjunto con el Instituto Avanzado de Tecnología de Samsung desarrollaron el robot humanoide Roboray mostrado en la Figura 1.4.

Figura 1.4: Robot Humanoide Roboray de Samsung [9]

Este robot es uno de los robots humanoides más avanzados en el mundo, cuenta con una altura de 140 cm y un peso de 50 kg. Posee una cámara estéreo en su cabeza y está compuesto por 53 actuadores diferentes incluyendo seis por cada pierna y 12 por cada mano. Lo más novedoso de su tecnología es que camina de una forma más humana al utilizar lo que se conoce como caminado dinámico. Esto significa que se encuentra en un estado de caída continua a cada paso y utiliza la gravedad para desplazarse reduciendo así el consumo de energía. Esta forma de caminado es novedosa ya que es similar a la forma en que caminan los humanos y debido a que la mayoría de los robots humanoides doblan sus rodillas para mantener su centro de masa bajo y estable. Una desventaja de utilizar este tipo de caminado es que introduce una dificultad mayor para los algoritmos de visión por computadora ya que los objetos en las imágenes se mueven más rápido. [9]

1.2. Planteamiento del problema

Son mucho los problemas de interés en la investigación sobre Robots Humanoides, entre los cuales destacan la percepción visual y el control con estabilidad de sus movimientos, en particular del caminado. Por otro lado, cabe destacar que las competiciones

entre Robots Humanoides en foros como RoboCup [10], FIRA [11] y TMR [12], entre otros muchos motivan a la investigación en estas áreas así como el desarrollo tecnológico correspondiente. Es así que en el DCA-CINVESTAV se ha incursionado en dichas competiciones desde el 2009, primero utilizando prototipos comerciales de pequeño formato y bajas prestaciones, como el Robonova I y el Bioloid, y luego usando prototipos con mejores prestaciones como el Nao y el DARwIn-OP. Para poder competir con los primeros ha sido necesario hacer modificaciones tanto en la estructura mecánica como en la computacional de los prototipos mencionados, sin conseguir obtener un robot competitivo hasta que se comenzó a utilizar el DARwIn-OP. Si bien los robots humanoides con los que cuenta el departamento, como DARwIn-OP y Nao, son muy eficientes en sus respectivos campos de aplicación, no son muy adecuados para soportar todo tipo de proyectos de investigación. Por ejemplo, el sistema computacional y de control del Nao es muy cerrado como para intentar proyectos de control de movimientos; mientras que el DARwIn-OP, siendo bastante más abierto, representa una estructura más orientada para las aplicaciones que para la investigación. Es así como surge la necesidad de contar con un robot humanoide cuya arquitectura computacional fuera completamente abierta a fin de tener el conocimiento y dominio de todos sus sistemas como el mecánico, el computacional y electrónico, el de control así como el de percepción incluyendo todos los sensores del robot.

1.3. Propósito del trabajo de Tesis

De este modo, el objetivo de la presente tesis es desarrollar un robot humanoide capaz de participar de manera competitiva en las competiciones como las organizadas por RoboCup, FIRA y TMR. Además, nos interesa que el prototipo desarrollado se convierta en una plataforma eficiente para soportar futuros proyectos en las áreas del departamento, como lo son la percepción (visual, inercial, táctil) y el control en lazo cerrado de todos sus movimientos. El prototipo propuesto se construyó a partir de su predecesor, el AH1N1, al cual se le hicieron modificaciones importantes en su estructura mecánica, sobre todo a nivel del tórax, cabeza y brazos. Además, se rediseñó el sistema

electrónico y computacional con base en el uso del controlador CM-700 de Robotis y el sistema Roboard-100 para el manejo de la algorítmica de percepción y control. En el marco de esta tesis se hizo el análisis cinemático y dinámico del nuevo prototipo, se implementaron algoritmos de visión estereoscópica simples capaces de cerrar el lazo de control en tareas de manipulación y se aplicaron técnicas de generación de patrones de marcha estable. Estas técnicas se validaron primero en simulación utilizando los modelos cinemático y dinámico obtenidos y se aplicaron posteriormente en tiempo real sobre la plataforma desarrollada.

1.4. Estructura del trabajo de Tesis

Al desarrollarse diversos sistemas para el prototipo, estos se introducen de manera individual en los capítulos siguientes pero también siguen una secuencia ya que los sistemas tienen elementos en común para finalmente integrarse.

- El Capítulo 2 presenta los sistemas mecánico, electrónico y computacional. Aquí se describe la estructura del prototipo anterior y las modificaciones que se hicieron para obtener el prototipo actual, así como los controladores y sensores incorporados al nuevo prototipo.

- El Capítulo 3 presenta el modelado cinemático, virtual y dinámico del humanoide. Aquí se presenta la cinemática, tanto directa como inversa, para las extremidades. Posteriormente se presenta el modelo geométrico usado para simular virtualmente al robot, el cual parte del modelo cinemático y por último la formulación de su modelo dinámico.

- El Capítulo 4 presenta el caminado. En este capítulo se presentan conceptos necesarios para comprender la generación de trayectorias en el espacio, así como una breve descripción de lo que involucra el caminado. Posteriormente se describen los métodos de caminado implementados y su análisis de estabilidad utilizando el criterio del ZMP.

- El Capítulo 5 presenta el sistema de percepción. En este capítulo se describen tanto el sistema de visión como su coordinación con los servomotores para alcance y manipulación de objetos.
- El Capítulo 6 presenta las pruebas y resultados de los algoritmos mencionados en los capítulos previos.
- Finalmente, en el Capítulo 7 se muestran las conclusiones obtenidas del trabajo de Tesis así como los objetivos alcanzados y el trabajo futuro con el prototipo.

Capítulo 2

Descripción del Robot Humanoide AH1N2

El Robot Humanoide AH1N2 se construyó a partir de su predecesor AH1N1 [13]. Este prototipo cuenta con 26 GDL (12 en Piernas, 10 en Brazos, 2 en Torso y 2 en Cabeza). Esto se traduce en 26 servomotores colocados directamente en cada articulación. Se utilizaron 2 servomotores RX-10 (cabeza), 4 servomotores RX-64 (torso y rodillas) y 22 servomotores RX-28 (piernas y brazos). Esto debido a que los pares requeridos varían, al igual que el peso de los servomotores. Los servomotores están unidos a partir de piezas estándar fabricadas por Robotis y piezas manufacturadas específicamente para el robot, estas piezas están hechas de Aluminio. Para el control de los servomotores se utilizó una Roboard RB-100 y el protocolo de comunicación RS-485. Se conectaron 2 cadenas, una para la parte superior y otra para la parte inferior, como se muestra en la Figura 2.1. Esto provocó que el bus de comunicación fuera muy largo y entonces se generaban problemas en la comunicación. Estas cadenas se conectaron en paralelo para poder conectarlas al único puerto RS-485 con el que cuenta la Roboard RB-100 y además para suministrarles el voltaje de manera independiente a la tarjeta.

Para el desarrollo del nuevo prototipo, primero se agregó una tarjeta como subcontrolador, dedicada exclusivamente a controlar los servomotores. Esta tarjeta es la CM-700. Al cambiar la tarjeta, también se cambiaron las conexiones entre los motores. Además, se cambió la estructura mecánica de la unidad pasajera y se agregó una cámara estereoscópica.

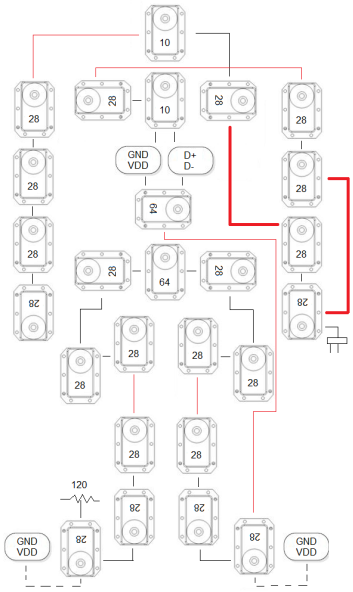


Figura 2.1: Conexón serie de 2 cadenas

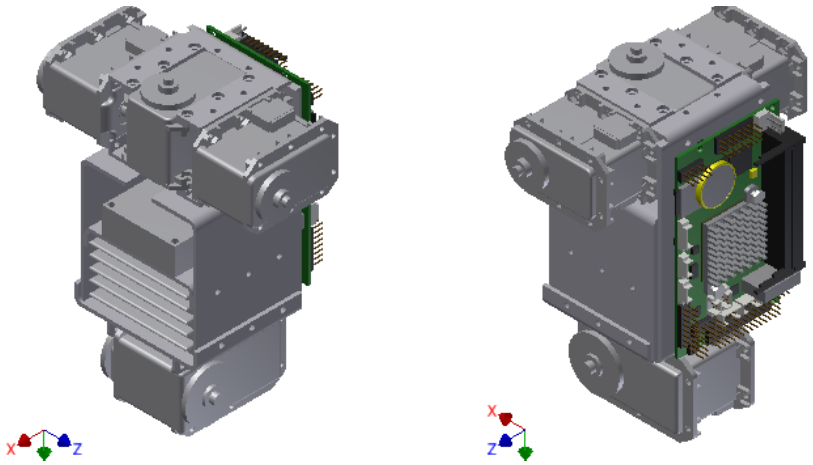
2.1. Sistema Mecánico

2.1.1. Modificaciones del nuevo diseño

El tórax del nuevo prototipo se rediseñó para incluir tanto el subcontrolador como las baterías, además de que en el nuevo prototipo, las tarjetas y las baterías se colocaron dentro del tórax.

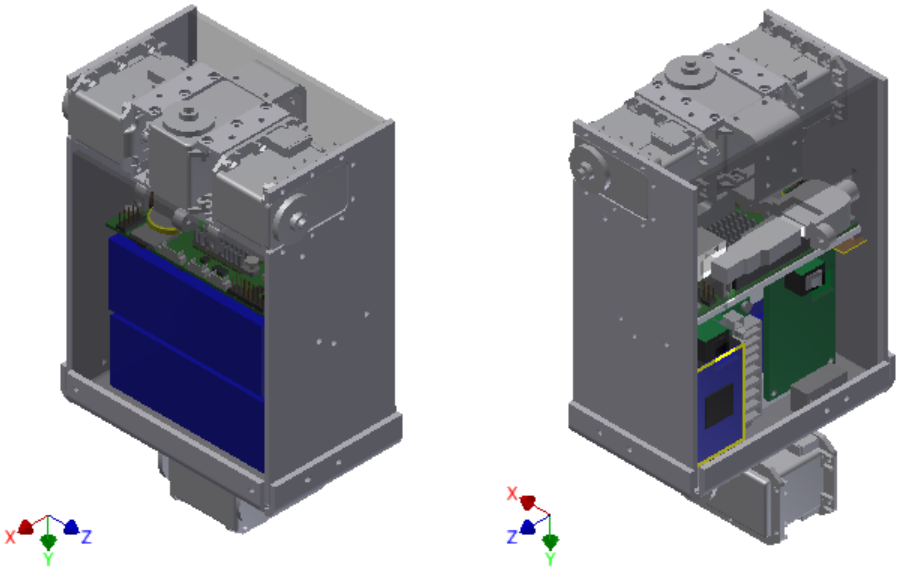
2.1.1.1. Tórax

El tórax del prototipo anterior mostrado en la Figura 2.2, fue diseñado para contener un banco de baterías y una unidad de medición inercial. La Roboard fue colocada en la pared posterior y los servomotores se sujetaron tanto a esta pared como a las paredes laterales. Este diseño carecía de protección para la Roboard en caso de una caída hacia atrás además de que el espacio en el tórax era muy reducido para incluir el subcontrolador. Es por esto que se optó por diseñar un tórax para el nuevo prototipo.



(a) Isométrico anterior (b) Isométrico posterior

Figura 2.2: Tórax del prototipo anterior



(a) Isométrico anterior (b) Isométrico posterior

Figura 2.3: Tórax del nuevo prototipo

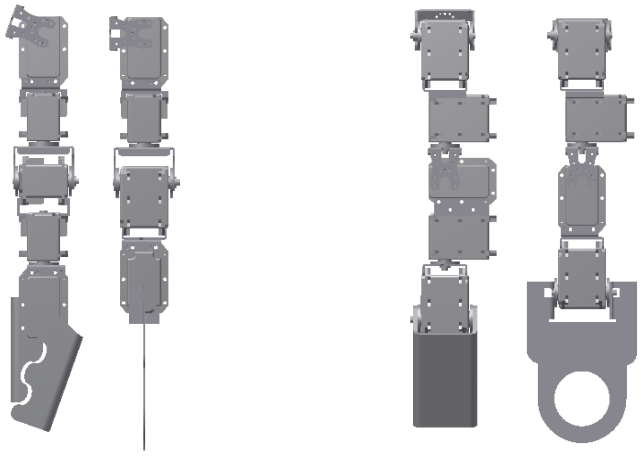
El tórax del nuevo prototipo mostrado en la Figura 2.3, cuenta con espacio para las 2 tarjetas además de 2 baterías Li-Po en vez del banco de baterías utilizado en el prototipo anterior. Además, cuenta con espacio para agregar una tarjeta adicional donde se añadirá la unidad de medición inercial además de dispositivos de entrada y

salida para la Roboard.

El sistema mecánico del prototipo anterior fue realizado a partir de los servomotores y de Duraluminio para proporcionar la rigidez necesaria a la estructura sin agregar peso de manera innecesaria. Para el nuevo prototipo se utilizó Alucobond en vez de Duraluminio ya que es más ligero y a la vez lo suficientemente rígido. También se utilizaron piezas impresas para los soportes de las tarjetas y como uniones para las placas del tórax. La configuración de las tarjetas fue elegida para que todos los puertos de ambas, estuviesen accesibles y a la vez para utilizar el menor espacio posible.

2.1.1.2. Brazos

El brazo del prototipo anterior contaba con 5 servomotores, incluyendo el servomotor encargado de abrir y cerrar el gripper. El brazo del nuevo prototipo cuenta con 6 debido a que se agregó el giro de la muñeca para aumentar su capacidad de manipular objetos. Ambos se muestran en la Figura 2.4. La configuración original de los servomotores también se modificó para la adición de un sexto servomotor, esto con el fin de que la longitud del brazo, tomando en cuenta únicamente los servomotores, aumentara ligeramente.

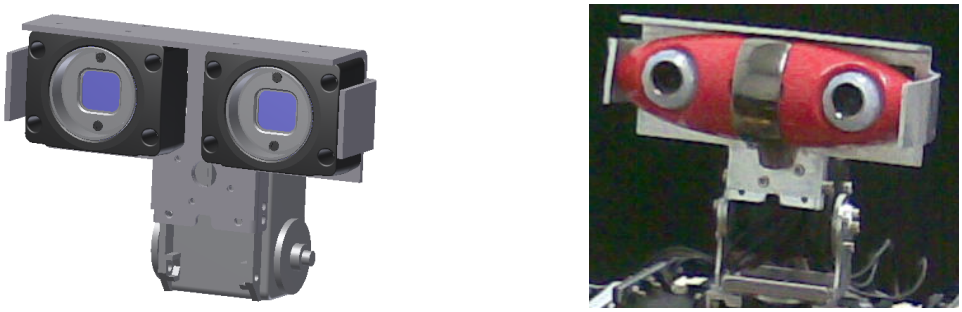


(a) Vista Frontal (b) Vista Lateral

Figura 2.4: Comparación de brazos de ambos prototipos

2.1.1.3. Cabeza

La cabeza del prototipo anterior mostrada en la Figura 2.5a, fue modificada ligeramente para sustituir las cámaras Firefly por la Minoru. Primero se realizó una perforación en la parte posterior para pasar el cable USB de la Minoru para que así quedara centrada. Una vez colocada, se ajustaron las paredes laterales para fijarla a la base de aluminio, como se muestra en la Figura 2.5b.



(a) Modelo anterior

(b) Prototipo actual

Figura 2.5: Comparación de cabezas de ambos prototipos

2.1.2. Servomotores

Como se observa en la Tabla 2.1, los servomotores RX-10 utilizan un voltaje menor que los otros 2 modelos, entonces se cambiaron los 2 servomotores RX-10 por RX-28 para poder aprovechar al máximo el rango de voltaje con el que trabajan los servomotores.

	RX-10	RX-28	RX-64
Par $(N \cdot m)$	1.3	3.7	5.3
Voltaje (V)	9 12	12 18.5	12 18.5
Corriente (A)	0.8	1.9	2.6

Tabla 2.1: Principales propiedades de los servomotores

Debido a que los servomotores se conectan en serie, previamente se asigna un ID a cada motor y este ID se utiliza dentro del paquete de datos donde se transmiten las direcciones de los datos que se escribirán a cada motor, como posición y velocidad.

2.2. Sistema Computacional

El motivo por el cual se decidió utilizar dos tarjetas fue incluir visión artificial en el robot humanoide, así como capacidades de más alto nivel como el control del caminado o de su comportamiento. La inclusión del segundo controlador cambió la manera en que se ejecutaron originalmente los algoritmos de caminado utilizando únicamente el controlador CM-700 o únicamente la Roboard. El cálculo de trayectorias y los programas de visión artificial se ejecutaron en la Roboard y el controlador se utilizó como interfaz para el manejo de los servomotores.

2.2.1. Roboard RB-100

La Roboard RB-100, mostrada en la Figura 2.6, es una computadora de 32 bits con un procesador de 1 GHz. Sus especificaciones completas se encuentran en el Apéndice A. Se utiliza como controlador principal donde se ejecutan los programas, tanto de visión como de trayectorias articulares y estos a su vez se comunican con el subcontrolador CM-700. Para trabajar con la Roboard, se le instaló la imagen de Lubuntu 10.04 disponible en la página web de Roboard. El proceso de configuración se encuentra descrito en el Apéndice B.

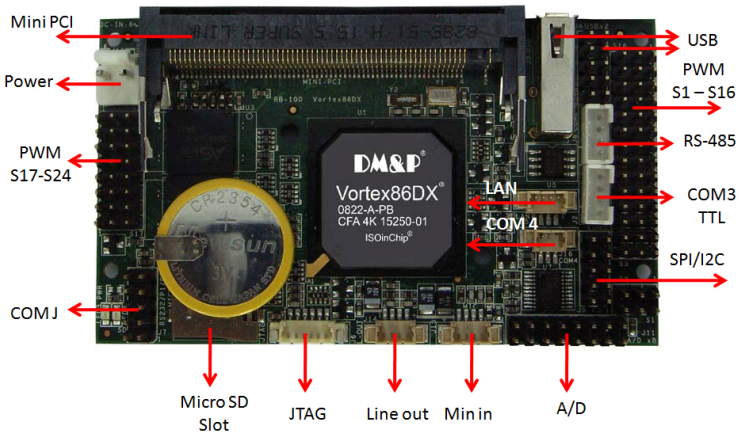


Figura 2.6: Roboard RB-100

2.2.2. CM-700

El controlador CM-700 mostrado en la Figura 2.7 se utiliza para controlar servomotores de Dynamixel a través de RS-485 (5 puertos) o TTL (4 puertos) dependiendo la serie de los motores utilizados. Tiene un procesador cuya frecuencia máxima de trabajo es de 16 MHz.

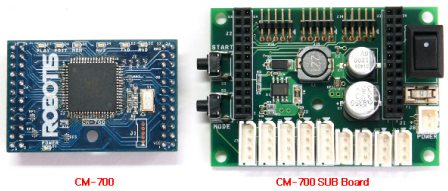


Figura 2.7: Controlador CM-700

Se conectaron los servomotores mediante 5 cadenas (Brazos, Piernas y una cadena Central), reemplazando las 2 cadenas previamente conectadas en el prototipo anterior. Esto con el fin de probar independientemente cada cadena cinemática sin la necesidad de conectar los 28 motores a la tarjeta, además de que se simplificaron las conexiones, como se muestra en la Figura 2.8.

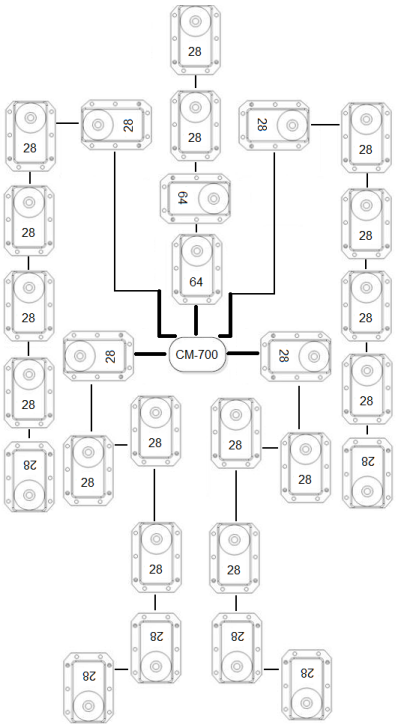


Figura 2.8: Conexiones de los servomotores al controlador CM-700

2.2.3. RoboPlus

RoboPlus es un programa distribuido por la empresa Robotis con el cual se prueban los servomotores y se cargan programas a los diferentes controladores de Robotis, en este caso el CM-700. Lo primero que se hace antes de trabajar con los servomotores de Dynamixel, es verificar su correcto funcionamiento y asignar sus correspondientes ID. Para transferir programas al controlador CM-700 se utiliza la Terminal del RoboPlus y el convertidor usb-serie LN-101. Estos programas no se crean directamente en el RoboPlus, a diferencia de los programas básicos con RoboPlus Task y RoboPlus Motion, estos se crean mediante Atmel Studio y se programan en lenguaje C.

2.2.4. Minoru 3D Webcam

El prototipo anterior utilizaba 2 cámaras Firefly mv de Point Grey. En este proyecto se optó por utilizar una cámara estereoscópica que utiliza un único puerto USB ya que las cámaras Point Grey requieren de controladores adicionales para utilizarse, además de que es más complicado utilizar 2 cámaras en OpenCV. La Minoru 3D Webcam, mostrada en la Figura 2.9, es una cámara diseñada para realizar conversaciones en línea desplegando imágenes 3D del usuario y debido a que es una cámara de uso comercial, fue más fácil utilizarla en conjunto con OpenCV.

Figura 2.9: Minoru 3D Webcam

2.2.5. Comunicación

La Roboard cuenta con diferentes protocolos de comunicación: RS-232, RS-485 y TTL, entre otros. Debido a la ausencia de documentación sobre el puerto serie del controlador CM-700 y a la incompatibilidad de voltajes entre los puertos de ambas tarjetas, no fue posible conectar directamente los puertos serie de los controladores. Así la comunicación entre los controladores se realizó mediante la comunicación serie utilizando el convertidor USB-serie LN-101. Utilizando este convertidor fue posible manipular el puerto USB de la Roboard como un puerto serie virtual y se logró la comunicación con el controlador CM-700 sin la necesidad de utilizar la terminal de RoboPlus, esto con el fin de controlar los servomotores desde la Roboard.

2.2.5.1. Roboard-CM

La comunicación serie entre las tarjetas se realiza a través de la transmisión y recepción de arreglos de caracteres sin signo, es por esto que, para comunicar la Roboard con el controlador CM-700, se desarrolló un algoritmo de codificación y decodificación de datos. Este algoritmo permite transmitir una variable de tipo doble utilizando únicamente dos datos de tipo caracter sin signo. Para cada servomotor se realizó lo siguiente:

1. Se toma la variable articular de la trayectoria y se asegura que no entre en conflicto con la transmisión de datos eliminando los ceros. Posteriormente se convierte a un arreglo de caracteres.

2. Se guarda su parte entera y dos decimales en arreglos separados.

3. Se convierten ambos arreglos de caracteres a entero y se guardan como caracteres sin signo, los cuales serán transmitidos. Al guardarse como caracteres sin signo, los valores quedan dentro del intervalo de 1 a 255 y la conversión de números negativos se realiza de manera automática.

El proceso inverso se utiliza para la decodificación dentro del controlador CM-700 y este proceso se ilustra en la Figura 2.10. El proceso detallado de conversión de datos se encuentra en el Apéndice D.

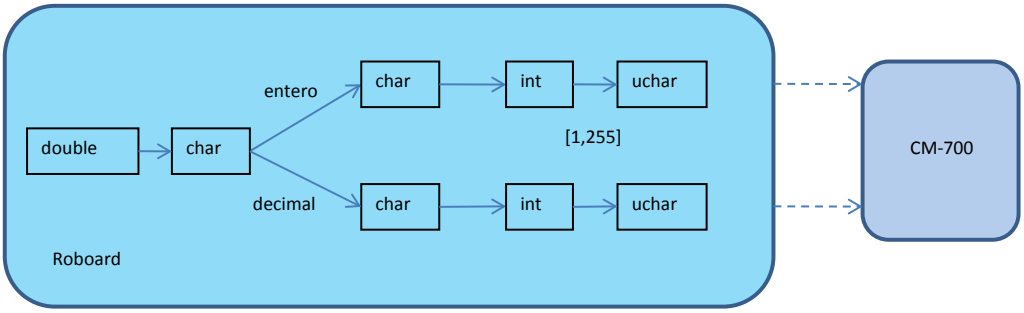


Figura 2.10: Conversión de datos para transmisión

Para mover los servomotores en conjunto, en vez de mandar posiciones de manera sincronizada a los 28 servomotores, se dividió el robot en cadenas cuyos servomotores se utilizan de manera sincronizada, ésto con el fin de reducir la cantidad de información transmitida. Para mover cada una de estas cadenas, se transmite primero un número que representa la cadena que se moverá y seguido se transmiten los ángulos para cada motor, como se muestra en la Figura 2.11. Las cadenas utilizadas son: pan, tilt y cadera; brazo derecho; brazo izquierdo y piernas. Sus números representativos son 1, 2, 3, 4, respectivamente.

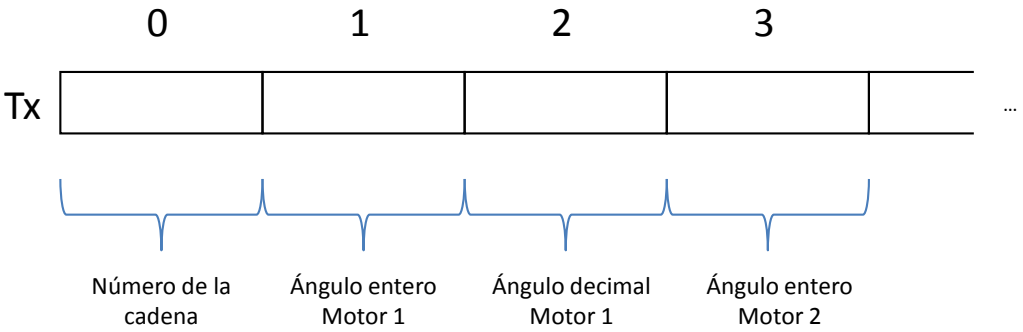


Figura 2.11: Arreglo de caracteres para transmisión

2.2.5.2. Roboard-Minoru

La comunicación con la cámara web desde la Roboard es muy sencilla debido al uso de Linux y OpenCV. Una vez conectada la cámara sólo se debe verificar que la detecte el sistema operativo en los registros de video y que se tengan los permisos suficientes

para accesar a ella. No se requiere instalar ningún controlador adicional y su manejo con OpenCV se hace como si se tuvieran conectadas dos cámaras web.

2.2.5.3. CM700-Servomotores

Para la comunicación con los servomotores desde el controlador CM-700, se utilizan las librerías de Dynamixel programadas en C Embebido. Para esto únicamente se utilizan dos instrucciones *dxl_read_word* y *dxl_write_word*. Ambas instrucciones reciben como primer parámetro el ID del servomotor y como segundo parámetro el registro en la memoria que se leerá o modificará, respectivamente. Los datos que utilizan son enteros de 2 bytes.

2.3. Sistema Electrónico

La Roboard se conectó en paralelo al controlador CM-700 para que ambos encendieran al mismo tiempo al igual que los servomotores. Es posible apagar el controlador una vez encendida la Roboard pero no se recomienda encenderlo, una vez encendida ésta. Esto debido al pico de corriente que los servomotores generan al momento de encenderse, ya que esto puede provocar que la Roboard se apague y se reinicie, y como cualquier otra computadora, lo cual no es deseable. Esta conexión en paralelo se realizó mediante la adición de una tercera tarjeta únicamente con un interruptor y tres conectores de dos vías, dos para la Roboard y para el controlador, y uno para la conexión paralela de las baterías. El consumo energético de los diversos componentes del prototipo, se muestran en la Tabla 2.2. A partir de estos datos y el tiempo de operación deseado, fueron seleccionaron las baterías.

	RB-100	CM-700	RX-28	RX-64	Total
RB-100	1				0.4
CM-700		1			0.04
RX-28			1		0.05
RX-64				1	0.05
Brazo	1	1	6		0.74
Pierna	1	1	5	1	0.74
Central	1	1	2	2	0.64
2 piernas	1	1	10	2	1.04
Completo	1	1	24	4	1.64

Tabla 2.2: Consumo energético en modo de espera

2.3.1. Baterías

Se utilizaron baterías Li-Po como fuente de voltaje ya que son los más utilizadas en sistemas embebidos debido a que su densidad de energía es mayor y por lo tanto su tamaño es reducido. Se utilizan dos baterías de 18.5 V y 2250 mAh conectadas en paralelo. las baterías a su vez se conectaron a una tarjeta donde se colocó un interruptor para encender ambos controladores.

Capítulo 3

Modelado

El modelado del sistema es fundamental para tratar de predecir su comportamiento bajo condiciones reales. Este puede ser cinemático o dinámico. El modelo cinemático se utiliza para obtener trayectorias en el espacio sin considerar las fuerza a las que estará sometido el prototipo. En cambio, el modelo dinámico considera las fuerzas necesarias para producir dicho movimiento y nos brinda una mejor aproximación a las condiciones reales en las que el prototipo se pondrá a prueba.

3.1. Modelo Cinemático

Para obtener el modelo cinemático del humanoide se utilizó la metodología de Denavit-Hartenberg. La cinemática de las piernas, tanto directa como inversa, fue resuelta utilizando esta metodología. Por su parte, la cinemática de los brazos fue resuelta mediante un método geométrico.

3.1.1. Metodología de Denavit-Hartenberg

La metodología de Denavit-Hartenberg (DH) se utiliza para determinar la pose (posición y orientación) del efector final con respecto a su base en función de los valores de sus coordenadas articulares (cinemática directa). Primero se establece un referencial fijo a cada eslabón del robot. Este referencial tiene su eje Z en el eje de revolución de la articulación que conecta el eslabón actual con el siguiente. Además, los referenciales asociados a eslabones adyacentes cumplen con las siguientes propiedades:

- DH1: El eje x_i es perpendicular al eje z_{i-1}

- DH2: El eje x_i interseca al eje z_{i-1}

Una vez fijos los referenciales, se hace una tabla cuyos renglones representan cada una de las articulaciones del mecanismo y cuyas columnas representan los parámetros DH $\theta_i, \alpha_i, a_i, d_i$

articulación i	θ_i	α_i	a_i	d_i
1				
2				
3				
$\vdots$				

Tabla 3.1: Tabla de parámetros de DH

donde:

- θ_i: ángulo entre x_{i-1} y x_i medido alrededor de z_{i-1}. θ_i es variable si la articulación i es rotacional.

- α_i: ángulo entre z_{i-1} y z_i medido alrededor del eje x_i.

- a_i: distancia sobre x_i desde O_i a la intersección de los ejes x_i y z_{i-1}.

- d_i: distancia sobre z_{i-1} desde O_i a la intersección de los ejes x_i y z_{i-1}. d_i es variable si la articulación i es prismática. [14]

El i-ésimo eslabón se caracteriza por su longitud a_i y su torcedura α_i, mientras que la i-ésima articulación se caracteriza por la distancia entre eslabones d_i y el ángulo entre eslabones θ_i. Cada uno de los renglones de esta tabla nos permite obtener las matrices A_i, donde cada matriz representa la transformación homogénea entre los referenciales $i-1$ e i, y tiene la forma siguiente:

$$A_i = \begin{bmatrix} R_{i-1}^i & d_{i-1}^i \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

donde $c_{\theta_i} = cos(\theta_i)$ y $c_{\theta_i} = sin(\theta_i)$. Una vez obtenidas las matrices $\{A_i : 1 \leq i \leq n\}$, se puede obtener la pose del referencial n con respecto al referencial 0 mediante la multiplicación de matrices

$$A_1 A_2 \cdots A_n = T_0^n$$

3.1.2. Cinemática de las piernas

3.1.2.1. Cinemática Directa

Para determinar la cinemática, primero se fijan los referenciales a los eslabones de la pierna de acuerdo a DH y se obtiene el diagrama cinemático mostrado en la Figura 3.1. Estos referenciales comienzan en la articulación de la cadera con el referencial 0 y terminan en el pie con el referencial 6. Una vez establecidos los referenciales, se obtienen los parámetros de DH mostrados en la Tabla 3.2

articulación i	θ_i	α_i	a_i	d_i
Jd1	θ_1	90°	0	0
Jd2	θ_2 - 90°	-90°	0	0
Jd3	θ_3	0	L_{d3}	0
Jd4	θ_4	0	L_{d4}	0
Jd5	θ_5	90°	0	0
Jd6	θ_6	0	L_{d5}	0

Tabla 3.2: Tabla de parámetros de la pierna

Utilizando la Tabla 3.2 se obtienen las matrices de paso A_i

$$A_1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} s_2 & 0 & c_2 & 0 \\ -c_2 & 0 & s_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} c_3 & -s_3 & 0 & L_{d3}c_3 \\ s_3 & c_3 & 0 & L_{d3}s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & L_{d4}c_4 \\ s_4 & c_4 & 0 & L_{d4}s_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5 = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6 = \begin{bmatrix} c_6 & -s_6 & 0 & L_{d5}c_6 \\ s_6 & c_6 & 0 & L_{d5}s_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

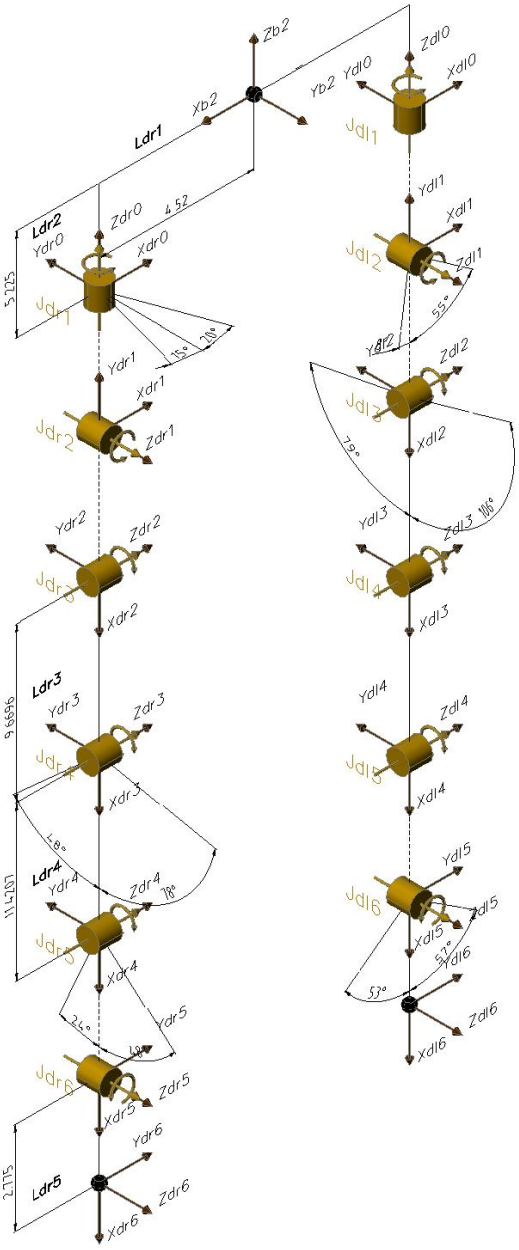


Figura 3.1: Diagrama Cinemático de las Piernas

Para obtener la transformación homogénea del referencial del pie con respecto a la articulación de la cadera realizamos la multiplicación de matrices

$$A_1A_2A_3A_4A_5A_6 = T_0^6 = \begin{bmatrix} x_6 & y_6 & z_6 & p_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

donde x_6, y_6 y z_6 representan los vectores unitarios del referencial 6. Los vectores [n,

s, a, p] representan el vector normal, el vector de deslizamiento (*sliding*), el vector de acercamiento (*approach*) y el vector de posición, respectivamente. Para obtener la transformación homogénea del referencial del pie con respecto al de la cadera (b2), incluimos la siguiente matriz de transformación:

$$A_0 = \begin{bmatrix} -1 & 0 & 0 & ls * L_{d1} \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -L_{d2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

donde *ls* es igual a 1 para la pierna derecha y -1 para la pierna izquierda.

$$A_0 A_1 A_2 A_3 A_4 A_5 A_6 = T_{b2}^6 \tag{3.3}$$

3.1.2.2. Cinemática Inversa

Para obtener la cinemática inversa de la pierna, es decir, los valores articulares a partir de una pose definida del pie, se utiliza el hecho de que los ejes de las primeras 3 articulaciones se intersecan. Se resuelve el problema cinemático inverso mediante la separación del problema en 2 partes: posición y orientación. El método de desacoplamiento cinemático establecido por Pieper [15] se utiliza para resolver la cinemática inversa de un robot de 6 gdl que tiene una muñeca esférica (3 ejes articulares que se intersecan) unida a su efector final, debido a esto la posición del efector final está dada por los primeros 3 gdl y su orientación por los últimos 3. Este método no puede aplicarse directamente ya que son las primeras 3 articulaciones las que se intersecan y no las últimas, debido a esto se debe utilizar la matriz de transformación inversa que relaciona las coordenadas de la cadera con el referencial del pie.

$$T_6^0 = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} n' & s' & a' & p' \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_6^{-1} A_5^{-1} A_4^{-1} A_3^{-1} A_2^{-1} A_1^{-1} \tag{3.4}$$

Una vez obtenida esta matriz, se aplica el método de desacoplamiento inverso [16], el cual consiste en determinar los ángulos θ_4, θ_5 y θ_6 a partir del vector de posición inverso y posteriormente obtener los ángulos restantes utilizando el método de transformación

inversa.

Para obtener θ_4, θ_5 y θ_6, igualamos los términos del vector de posición p' en ambos lados de la ecuación (3.4)

$$-C_6(C_{45}L_{d3} + C_5L_{d4}) = p'_x + L_{d5} \tag{3.5}$$

$$-S_{45}L_{d3} - S_5L_{d4} = p'_z \tag{3.6}$$

$$S_6(C_{45}L_{d3} + C_5L_{d4}) = p'_y \tag{3.7}$$

donde $C_{ij} = cos(\theta_i + \theta_j)$ y $S_{ij} = sin(\theta_i + \theta_j)$.

Elevando al cuadrado y sumando las 3 ecuaciones previas, obtenemos C_4

$$C_4 = \frac{(p'_x + L_{d5})^2 + p'^2_y + p'^2_z - L^2_{d3} - L^2_{d4}}{2L_{d3}L_{d4}}$$

Utilizamos la función *atan2* para obtener θ_4

$$\theta_4 = atan2(\pm\sqrt{1 - C_4^2}, C_4) \tag{3.8}$$

Elevamos al cuadrado, sumamos las ecuaciones (3.5) y (3.6), y expandimos el resultado para obtener

$$C_5(C_4L_{d3} + L_{d4}) - S_4S_5L_{d3} = \pm\sqrt{(p'_x + L_{d5})^2 + p'^2_y} \tag{3.9}$$

Expandimos la ecuación (3.7) y obtenemos

$$S_6(C_4L_{d3} + L_{d4}) + C_5(S_4L_{d3}) = -p'_z \tag{3.10}$$

Sean $C_4L_{d3} + L_{d4} = rC_\psi$ y $S_4L_{d3} = rS_\psi$, y al sustituir los términos en las ecuaciones (3.9) y (3.10), obtenemos correspondientemente

$$rC_{5\psi} = \pm\sqrt{(p'_x + L_{d5})^2 + p'^2_y} \tag{3.11}$$

$$rS_{5\psi} = -p'_z \tag{3.12}$$

donde $r = \sqrt{(p'_x + L_{d5})^2 + p'^2_y + p'^2_z}$ y $\psi = atan2(S_4L_{d3}, C_4L_{d3} + L_{d4})$. Dividimos la

ecuación (3.12) por la ecuación (3.11), obtenemos $tan(\theta_5 + \psi)$ y finalmente

$$\theta_5 = atan2(-p'_z, \pm\sqrt{(p'_x + L_{d5})^2 + p'^2_y}) - \psi \tag{3.13}$$

Dividimos la ecuación (3.6) por la ecuación (3.5) y obtenemos θ_6

$$\theta_6 = atan2(p'_y, -p'_x - L_{d5}) \tag{3.14}$$

Para aplicar el método de transformación inversa y obtener las variables restantes, primero premultiplicamos la ecuación (3.4) por A_6

$$A_6 \begin{bmatrix} n' & s' & a' & p' \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_5^{-1} A_4^{-1} A_3^{-1} A_2^{-1} A_1^{-1} \tag{3.15}$$

Definimos la matriz del lado izquierdo como G_2^{LHS} y la matriz del lado derecho como G_2^{RHS}

$$G_2^{LHS} = A_6 \begin{bmatrix} n' & s' & a' & p' \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} n'_x C_6 - n'_y S_6 & s'_x C_6 - s'_y S_6 & a'_x C_6 - a'_y S_6 & p'_x C_6 - p'_y S_6 + L_{d5} C_6 \\ n'_x S_6 + n'_y C_6 & s'_x S_6 + s'_y C_6 & a'_x S_6 + a'_y C_6 & p'_x S_6 + p'_y C_6 + L_{d5} S_6 \\ n'_z & s'_z & a'_z & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G_2^{RHS} = A_5^{-1} A_4^{-1} A_3^{-1} A_2^{-1} A_1^{-1} =$$

$$\begin{bmatrix} C_1 S_2 C_{345} - S_1 S_{345} & S_1 S_2 C_{345} + C_1 S_{345} & -C_2 C_{345} & -L_{d3} C_{45} - L_{d4} C_5 \\ C_1 C_2 & S_1 C_2 & S_2 & 0 \\ C_1 S_2 S_{345} + S_1 C_{345} & S_1 S_2 S_{345} - C_1 C_{345} & -C_2 S_{345} & -L_{d3} S_{45} - L_{d4} S_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Comparamos el elemento (2,3) de G_2^{RHS} y G_2^{LHS} y obtenemos θ_2

$$\theta_2 = atan2(a'_x S_6 + a'_y C_6, \pm\sqrt{1 - (a'_x S_6 + a'_y C_6)^2}) \tag{3.16}$$

Si comparamos los elementos (2,1) y (2,2), obtenemos 2 ecuaciones

$$C_1C_2 = n'_xS_6 + n'_yC_6 \tag{3.17}$$

$$S_1C_2 = s'_xS_6 + s'_yC_6 \tag{3.18}$$

Dividiendo la ecuación (3.18) por la ecuación (3.17) obtenemos θ_1

$$\theta_1 = atan2(s'_xS_6 + s'_yC_6, n'_xS_6 + n'_yC_6) \tag{3.19}$$

Si comparamos los elementos (1,3) y (3,3), obtenemos 2 ecuaciones

$$-C_2C_{345} = a'_xC_6 - a'_yS_6 \tag{3.20}$$

$$-C_2S_{345} = a'_z \tag{3.21}$$

Dividiendo la ecuación (3.21) por la ecuación (3.20) obtenemos θ_{345}

$$\theta_{345} = atan2(a'_z, a'_xC_6 - a'_yS_6) \tag{3.22}$$

Finalmente, obtenemos θ_3

$$\theta_3 = \theta_{345} - \theta_4 - \theta_5 \tag{3.23}$$

3.1.3. Cinemática de los Brazos

3.1.3.1. Cinemática Directa

Para determinar la cinemática se realiza un proceso similar al de las piernas. Primero se obtiene el diagrama cinemático mostrado en la Figura 3.2. Estos referenciales comienzan en el hombro con el referencial 0 y terminan en el centro de la mano con el referencial 5. Una vez establecidos los referenciales, se obtienen los parámetros de DH mostrados en la Tabla 3.3

articulación i	θ_i	α_i	a_i	d_i
Ju1	$\theta_1 - 90$	90°	0	0
Ju2	$\theta_2 + 90°$	-90°	0	0
Ju3	$\theta_3 + 90$	90°	0	$-L_{u2}$
Ju4	θ_4	-90°	0	0
Ju5	θ_5	0°	0	$-L_{u3}$

Tabla 3.3: Tabla de parámetros del brazo

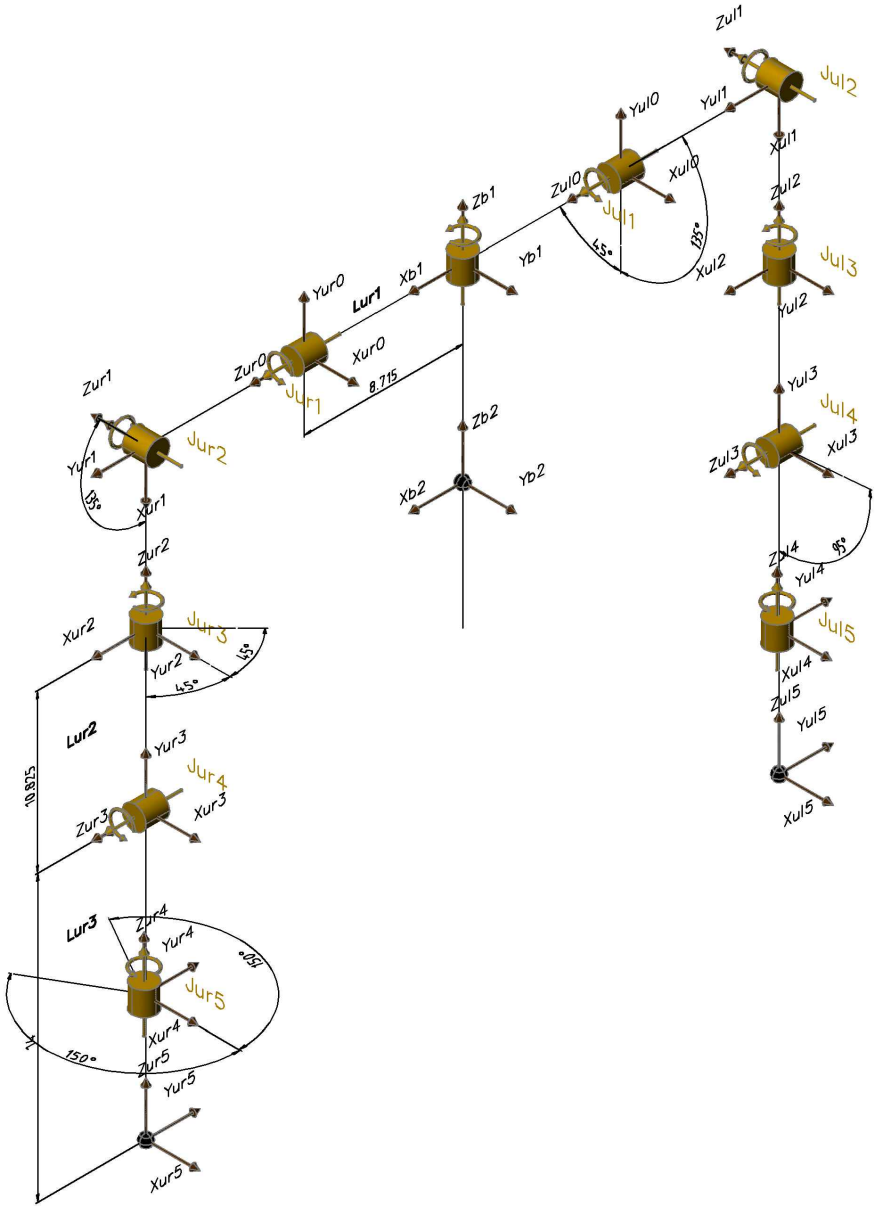


Figura 3.2: Diagrama cinemático de los brazos

Utilizando la Tabla 3.3 obtenemos las matrices de paso A_i

$$A_1 = \begin{bmatrix} s_1 & 0 & -c_1 & 0 \\ -c_1 & 0 & -s_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} -s_2 & 0 & -c_2 & 0 \\ c_2 & 0 & -s_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} -s_3 & 0 & c_3 & 0 \\ c_3 & 0 & s_3 & 0 \\ 0 & 1 & 0 & -L_{u2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & -L_{u3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para obtener la transformación homogénea del referencial de la mano con respecto al del hombro realizamos la multiplicación de matrices

$$A_1 A_2 A_3 A_4 A_5 = T_0^5 = \begin{bmatrix} x_5 & y_5 & z_5 & p_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

Para obtener la transformación homogénea del referencial de la mano con respecto al del pecho (b1), incluimos la siguiente matriz de transformación:

$$A_0 = \begin{bmatrix} 0 & 0 & 1 & ls * L_{u1} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

donde *ls* es igual a 1 para el brazo derecho y -1 para el brazo izquierdo.

$$A_0 A_1 A_2 A_3 A_4 A_5 = T_{b1}^5 \tag{3.26}$$

3.1.3.2. Cinemática Inversa

Para resolver la cinemática inversa de los brazos no es posible utilizar la cinemática directa ya que, debido a que tienen 5 gdl, no es posible especificar una orientación independiente de la posición en todo el espacio de trabajo. Debido a esto, se optó por utilizar un método geométrico que dependiera únicamente de la posición para determi-

nar la cinemática inversa. Ya que el giro de la muñeca no interviene en la posición de la mano, este gdl no se considera para la cinemática inversa.

Primero expresamos las coordenadas de la mano con respecto al referencial del hombro (X_0, Y_0, Z_0) y se obtiene la flexión-extensión del codo, representado por θ_4, de manera exacta a partir del triángulo que se forma entre el brazo y el vector que va del hombro a la mano, mostrado en la Figura 3.3.

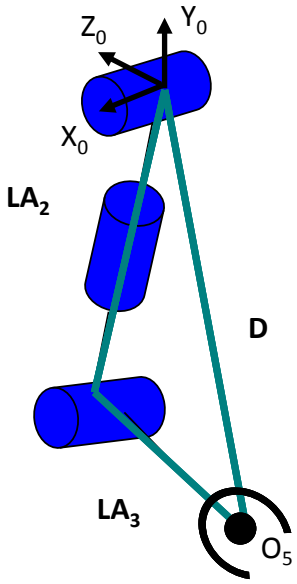


Figura 3.3: Triángulo formado por el brazo y el vector de posición

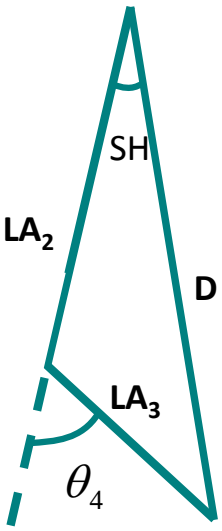


Figura 3.4: Triángulo con flexión-extensión del codo y SH

Tomando el plano en el que se encuentra este triángulo, mostrado en la Figura 3.4, se obtiene la variable articular θ_4 utilizando ley de cosenos.

$$C_4 = \frac{L_{A2}^2 + L_{A3}^2 - ||\mathbf{D}||^2}{2L_{A2}L_{A3}} \tag{3.27}$$

$$\theta_4 = atan2(-\sqrt{1 - C_4^2}, C_4) \tag{3.28}$$

Para obtener θ_1 utilizamos el ángulo denominado SH (ángulo entre el brazo y el vector $\mathbf{D}$) y la magnitud del vector $\mathbf{D}$ proyectado en el plano sagital, mostrado en la Figura 3.5. Obtenemos un vector $\mathbf{D_{eq}}$ y con la magnitud del vector $\mathbf{D}$ original obtenemos un factor de escala.

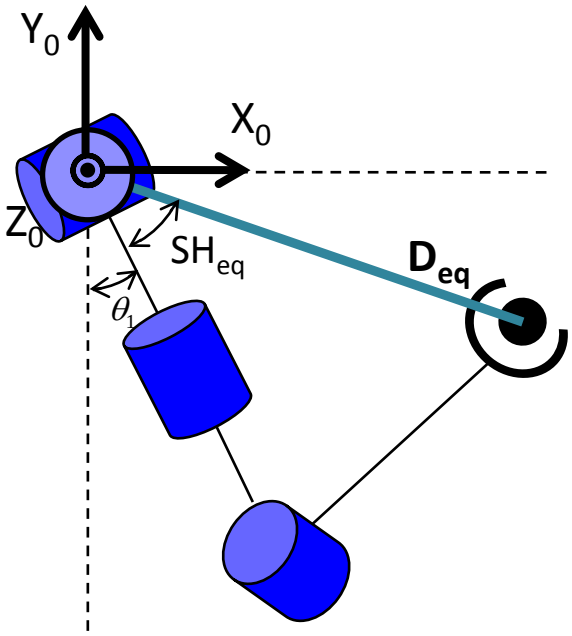


Figura 3.5: Proyección en el plano sagital

$$C_{SH} = \frac{L_{A2}^2 + ||\mathbf{D}||^2 - L_{A3}^2}{2||\mathbf{D}||L_{A2}} \tag{3.29}$$

$$SH = acos(C_{SH}) \tag{3.30}$$

$$\mathbf{D_{eq}} = \sqrt{\mathbf{D}_x^2 + \mathbf{D}_y^2 + 0} \tag{3.31}$$

$$Scf = \frac{||\mathbf{D_{eq}}||}{||\mathbf{D}||} \tag{3.32}$$

$$\mathbf{SH_{eq}} = \mathbf{Scf} * \mathbf{SH} \tag{3.33}$$

Si las coordenadas Y y Z son 0, el cálculo de θ_1 se reduce a

$$\theta_1 = 90 - atan2(-\mathbf{D}_y, \mathbf{D}_x) - SH. \tag{3.34}$$

Cuando las coordenadas son diferentes de 0, se presentan 2 casos: Cuando la coordenada y es negativa y cuando es positiva. Cuando la coordenada es negativa, la variable articular θ_1 se calcula como

$$\theta_1 = 90 - atan2(-\mathbf{D}_y, \mathbf{D}_x) - SH_{eq}. \tag{3.35}$$

Cuando la coordenada y es positiva, la variable articular θ_1 se calcula como

$$\theta_1 = 90 - atan2(-\mathbf{D}_y, \mathbf{D}_x) - \frac{SH}{Scf}. \tag{3.36}$$

Para el cálculo de θ_2 se tienen 2 casos: Cuando el producto de la coordenada z con el signo del brazo es menor o igual que 0 y cuando es mayor que 0

$$L * z <= 0 \tag{3.37}$$

donde L es igual a 1 cuando el consideramos el brazo derecho y L es igual a -1 cuando consideramos el brazo izquierdo.

Cuando el producto es menor o igual que 0, simplemente se fija θ_2 como 0 y cuando es mayor que 0, se toma directamente el ángulo formado entre las componentes x y z del vector $\mathbf{D}$, mostrado en la Figura 3.6

$$\theta_2 = atan2(\mathbf{D}_z, \mathbf{D}_x) \tag{3.38}$$

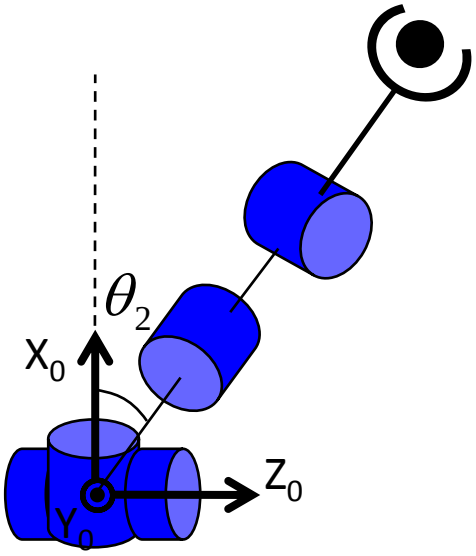


Figura 3.6: Cálculo de θ_2

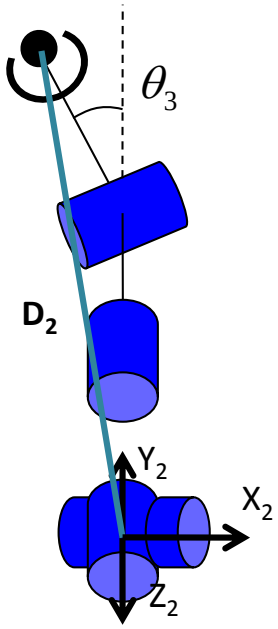


Figura 3.7: Cálculo de θ_3

Una vez obtenidos θ_1 y θ_2, calculamos la posición de la mano con respecto a O_2 y

calculamos θ_3 a partir de sus componentes x e y, mostrados en la Figura 3.7.

$$T_0^2 = A_1 A_2 \tag{3.39}$$

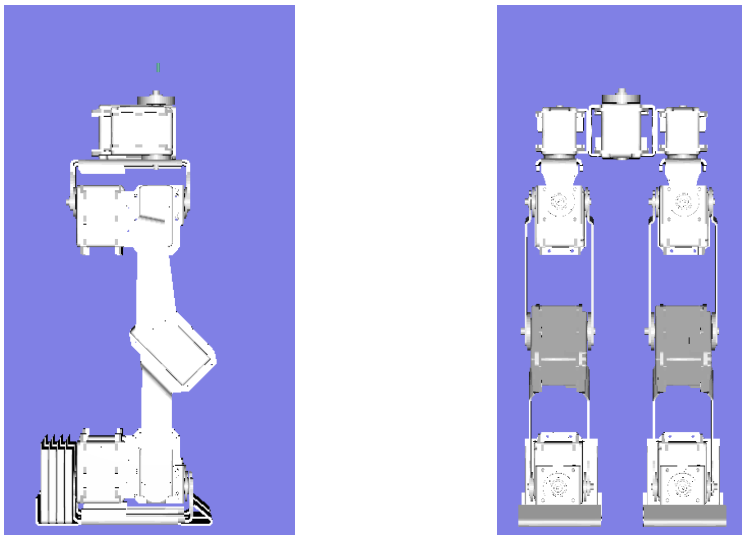
$$\mathbf{D}_2 = (T_0^2)^{-1}\mathbf{D} \tag{3.40}$$

$$\theta_3 = atan2(-\mathbf{D}_{2x}, \mathbf{D}_{2y}) \tag{3.41}$$

Este valor de θ_3 se calcula para ajustar la posición y reducir el error.

3.2. Modelado Virtual

Para el modelo virtual del humanoide se utilizó el lenguaje VRML (Virtual Reality Modeling Language), este lenguaje permite crear mundos virtuales y se puede utilizar en conjunto con diversos lenguajes de programación, en particular con Matlab. El uso de esta herramienta es únicamente como ayuda visual para la comprobación de la cinemática y para la generación de trayectorias. Esto quiere decir que los programas realizados no dependen de realimentación por parte del modelo virtual.



(a) Vista Lateral (b) Vista Frontal

Figura 3.8: Modelo VRML de la Unidad Locomotora

3.2.1. Unidad Locomotora

Para comprobar la cinemática inversa de las piernas, primero se construyó un modelo virtual que consiste en ambas piernas y la cadera, mostrado en la Figura 3.10.

Este modelo virtual se construye a partir de los modelos virtuales creados mediante Inventor y posteriormente se ensamblan de manera jerárquica tomando la cadera como base y los demás eslabones se ensamblan utilizando el diagrama de DH como referencia. Primero, se transforman los referenciales de cada eslabón para que coincida el eje Z de los referenciales de DH y el eje Y del mundo virtual. Posteriormente se ensamblan 2 cadenas cinemáticas abiertas partiendo de la cadera cuyos efectores finales son ambos pies. Este proceso de ensamble se encuentra de manera detallada en la referencia [17] y el código generado en el Apéndice E. El modelo virtual puede manipularse mediante la interacción con un usuario o mediante otros lenguajes de programación.

3.2.1.1. Simulador Cinemático

Para la comprobación de la cinemática, tanto directa como inversa, se utilizó Matlab. Este software tiene la ventaja de que tiene librerías para conectarse directamente con archivos VRML a través de interfaces gráficas o incluso a través de Simulink. Se eligió desarrollar una interfaz gráfica conectada a un mundo virtual para comprobar tanto cinemática inversa como cinemática directa, esta interfaz se muestra en la Figura 3.9.

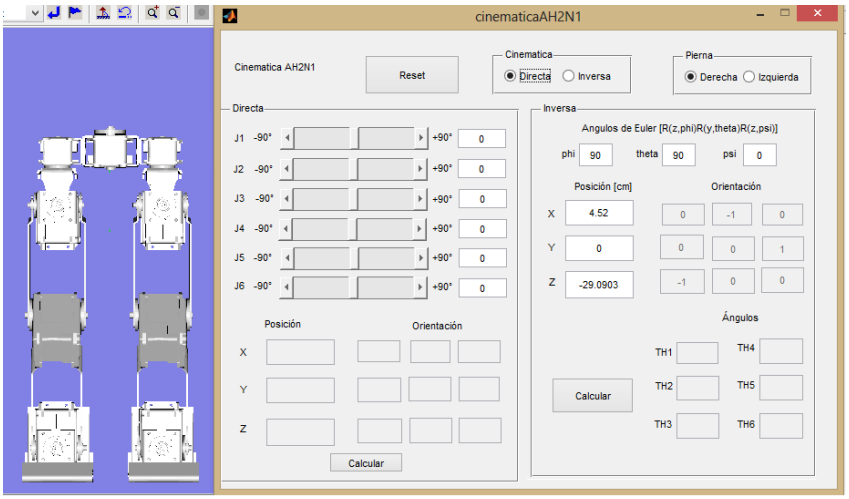
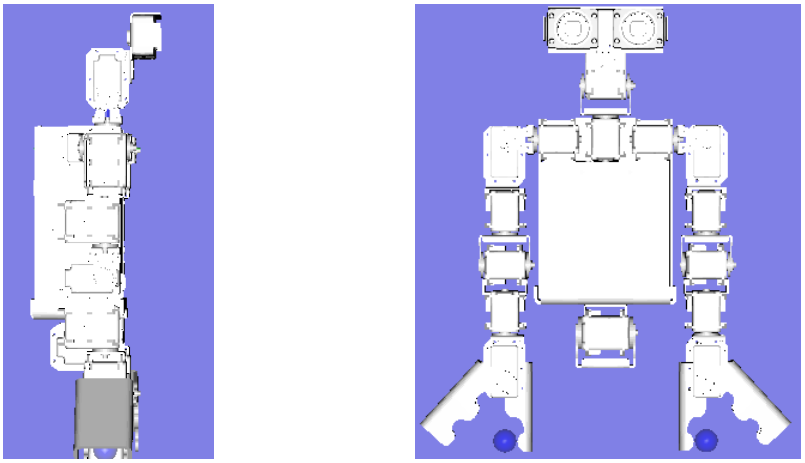


Figura 3.9: Interfaz de la unidad locomotora

3.2.2. Unidad Pasajera

Para comprobar la cinemática inversa de los brazos, se construyó el modelo virtual que incluye el tórax, los brazos y la cabeza, mostrado en la Figura 3.10.



(a) Vista Lateral (b) Vista Frontal

Figura 3.10: Modelo VRML de la Unidad Pasajera

Este modelo se construye de la misma manera que el anterior, utilizando el tórax como base y conectándole 3 cadenas cinemáticas abiertas. Además incluye 2 puntos de vista en los ojos, mostrados en la Figura 3.11 , los cuales permiten simular visión artificial dentro del mundo virtual.

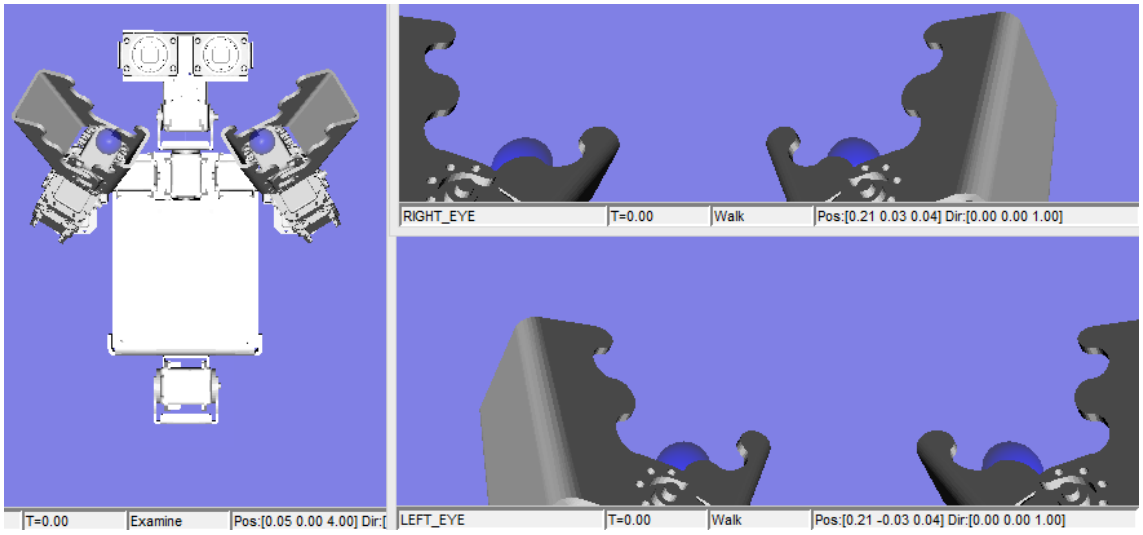


Figura 3.11: Puntos de vista de las cámaras

3.2.2.1. Simulador Cinemático

Para la comprobación de la cinemática, tanto directa como inversa, se desarrolló una interfaz gráfica así como la mencionada anteriormente, esta interfaz se muestra en la Figura 3.12.

Figura 3.12: Interfaz de la unidad pasajera

3.2.3. Modelo Completo

Para el modelo completo se exportó la unidad pasajera completa a VRML y se ensambló con la unidad locomotora. Este modelo se muestra en la Figura 3.13. No se utilizó el modelo de la unidad pasajera ya que el modelo completó se utilizó con el fin de visualizar las trayectorias de caminado además de que el costo computacional se incrementa con el número de transformaciones dentro de los archivos VRML. Este modelo además se colocó dentro de una cancha de fútbol, la cual proporciona una referencia para visualizar el movimiento del robot además de proporcionar un fondo al modelo virtual.

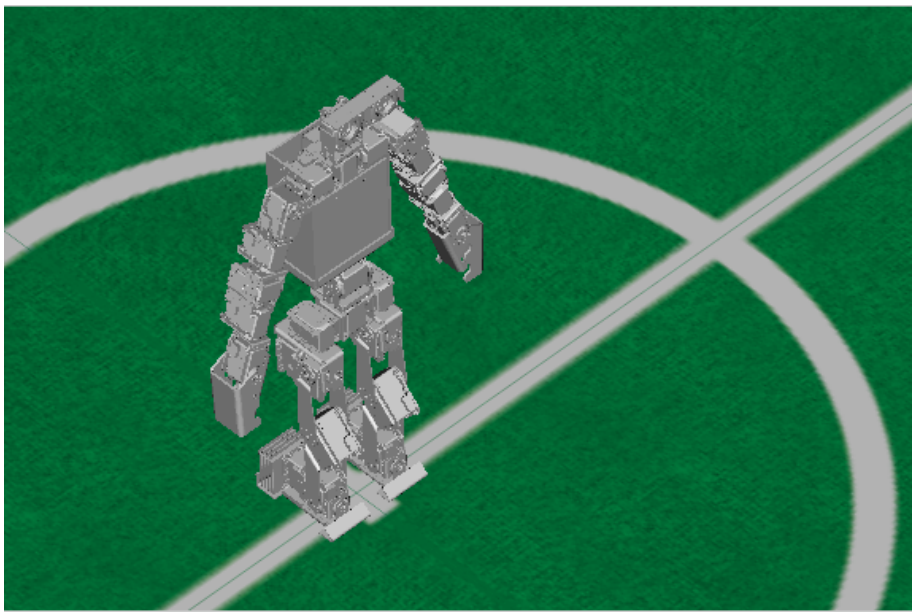


Figura 3.13: Modelo VRML completo

3.3. Modelo Dinámico

Para el modelo dinámico del humanoide se utilizó la formulación de Newton-Euler debido a que su costo computacional es menor que el de la formulación de Euler-Lagrange. Debido al número de gdl y a que el movimiento de la unidad pasajera es menor comparado con el de la unidad locomotora, se utilizó un modelo dinámico con menos grados de libertad. Este modelo consiste de 12 gdl y considera la unidad pasajera como un eslabón en conjunto con la cadera.

3.3.1. Formulación Newton-Euler

La formulación de Newton-Euler es un método utilizado para analizar la dinámica de robots manipuladores. En este método se trata cada eslabón del robot por separado y se obtienen las ecuaciones que describen su momento linear y su movimiento angular. Ya que cada eslabón está acoplado a otros, las ecuaciones contienen fuerzas acopladas y pares que aparecen también en las ecuaciones que describen eslabones vecinos. La formulación de Newton-Euler es superior a la formulación Lagrangiana cuando se desea conocer las fuerzas generalizadas necesitan aplicarse con el fin de realizar una secuencia

particular de coordenadas generalizadas en el tiempo. Esto es útil cuando no nos interesa conocer la relación funcional general entre ambas. Además, el computo iterativo del método permite reducir el tiempo de cálculo.

Para obtener las ecuaciones de movimiento para un manipulador de n-ecuaciones, primero escogemos referenciales 0,...,n, donde el referencial 0 es un referencial inercial, y el referencial i está fijo al eslabón i para $i \geq 1$. Ahora presentamos diversos vectores expresados en el referencial i. El primer conjunto de vectores pertenece a las velocidades y las aceleraciones de diferentes partes del manipulador. [14]

- a_i^G = aceleración del centro de masa del eslabón i
- a_i^i = aceleración lineal del eslabón expresada en el referencial i
- ω_i^i = velocidad angular del eslabón expresada en el referencial i
- α_i = aceleración angular del eslabón expresada en el referencial i

El siguiente conjunto de vectores corresponde a fuerzas y pares.

- $\boldsymbol{g}_0$ = aceleración debida a la gravedad (expresada en el referencial *0*)
- $f_i^{i,i-1}$ = fuerza que el eslabón *i-1* ejerce sobre el eslabón i expresada en el referencial i
- τ_i = par que el actuador de la articulación i-ésima debe ejercer
- R_i^{i-1} = matriz de rotación del referencial i al referencial *i-1*

EL conjunto final de vectores pertenece a características físicas del manipulador.

- m_i = masa del eslabón i
- I_{G_i} = matriz de inercia del eslabón i expresado en un referencial paralelo al referencial i, cuyo origen está en el centro de masa del eslabón i
- $\boldsymbol{r}_i^{G/i}$ = vector de la articulación i al centro de masa del eslabón i expresado en el referencial i

- $\boldsymbol{r}_i^{i/i-1}$ = *vector de la articulación i-1 a la articulación i expresado en el referencial i*

- ρ = *tipo de articulación (1 si es rotacional, 0 si es prismática)*

Para cada eslabón se calcula lo siguiente [18]:

1. Velocidad angular

$$\omega_i^i = R_i^{i-1}(\omega_{i-1}^{i-1} + \rho_i \dot{q}_i \boldsymbol{z}) \tag{3.42}$$

2. Aceleración angular

$$\alpha_i^i = R_i^{i-1}(\alpha_{i-1}^{i-1} + \rho_i \ddot{q}_i \boldsymbol{z} + \omega_{i-1}^{i-1} \times \rho_i \dot{q}_i \boldsymbol{z}) \tag{3.43}$$

3. Aceleración lineal del eslabón

$$a_i^i = R_i^{i-1} a_{i-1}^{i-1} + \alpha_i^i \times \boldsymbol{r}_i^{i/i-1} + \omega_i^i \times (\omega_i^i \times \boldsymbol{r}_i^{i/i-1}) \tag{3.44}$$

4. Aceleración del centro de masa de cada eslabón

$$a_i^G = a_i^i + \alpha_i^i \times \boldsymbol{r}_i^{G/i} + \omega_i^i \times (\omega_i^i \times \boldsymbol{r}_i^{G/i}) \tag{3.45}$$

5. Fuerza incercial de cada eslabón

$$F_i = m_i a_i^G \tag{3.46}$$

6. Momento inercial de cada eslabón

$$M_{Gi} = I_{G_i} \alpha_i^i + \omega_i^i \times I_{G_i} \omega_i^i \tag{3.47}$$

7. Ecuación de Newton

$$f_i^{i,i-1} = f_i^{i+1,i} - m_i R_i^0 \boldsymbol{g}_0 + F_i \tag{3.48}$$

8. Ecuación de Euler

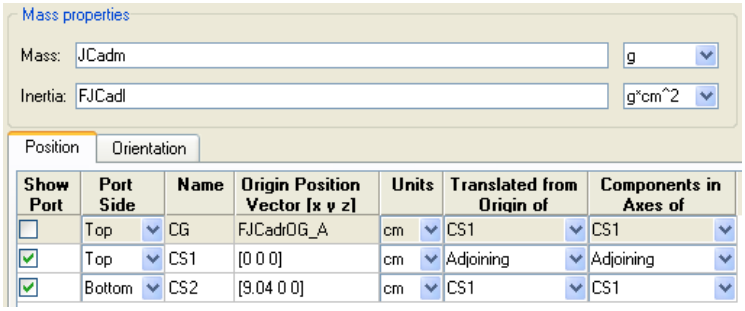
$$n_i^{i,i-1} = n_i^{i+1,i} + (\boldsymbol{r}_i^{i/i-1} + \boldsymbol{r}_i^{G/i}) \times \boldsymbol{f}_i^{i,i-1} - \boldsymbol{r}_i^{G/i} \times \boldsymbol{r}_i^{i+1,i} + M_{Gi} \tag{3.49}$$

Ya que este algoritmo está diseñado para una cadena cinemática abierta simple y el humanoide es arborescente, debemos considerar al humanoide como una cadena cinemática abierta de 12 gdl, cuya base es el pie de apoyo y cuyo efector final el pie flotante. La fase de soporte doble se considera instantánea y, por lo tanto, simplemente se invierte el orden de la cadena. A partir de las trayectorias de las articulaciones, se calcula de manera instantánea los pares necesarios para producir este movimiento. El programa desarrollado se puede encontrar en el Apéndice F.3.

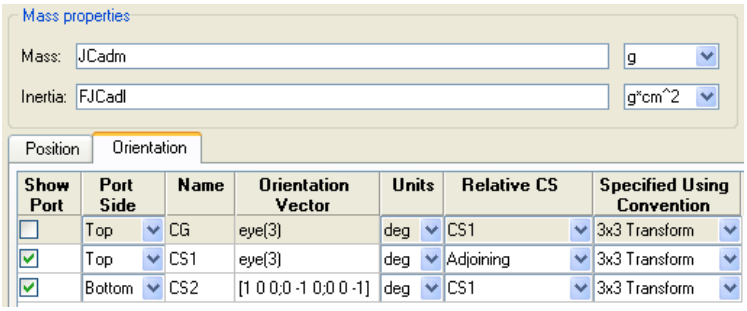
3.3.2. SimMechanics

Para comprobar la validez del modelo desarrollado en la sección anterior, se desarrolló un modelo dinámico utilizando la caja de herramientas *SimMechanics* de *Simulink*. Esta caja de herramientas permite hacer simulaciones dinámicas de máquinas con múltiples eslabones articulados. Esto se puede realizar de dos maneras: utilizando trayectorias de movimiento como entrada al modelo o trayectorias de fuerza. En este caso se utilizaron las trayectorias de movimiento que ya fueron generadas a partir de los algoritmos de caminado. Una vez implementadas las trayectorias, el modelo permite añadir sensores a las articulaciones y así poder observar los pares necesarios en las articulaciones para producir el movimiento deseado.

Para hacer uso de esta caja, primero se especifican las propiedades de cada eslabón dentro de un bloque *Body*, mostrado en la Figura 3.14. Dentro de este bloque, se especifican su tensor de inercia y su masa. Además, se debe especificar la posición del centro de masa, así como la pose de las articulaciones correspondientes a cada eslabón. Este se hace fijando dos referenciales: uno en la articulación anterior, el cual está fijo al eslabón actual y un referencial en la siguiente articulación, el cual se encuentra fijo al eslabón siguiente. Esto difiere de la metodología de DH debido a que el primer referencial de la cadena cinemática se encuentra fijo a la base del pie, y no a la primera articulación. Cabe aclarar que, tanto el referencial siguiente como el centro de masa, se describen a partir del referencial fijo al eslabón. La orientación se puede especificar como una matriz de rotación, un cuaternión o por cualquier combinación de ángulos de Euler.



(a) Posición



(b) Orientación

Figura 3.14: Propiedades del eslabón Cadera

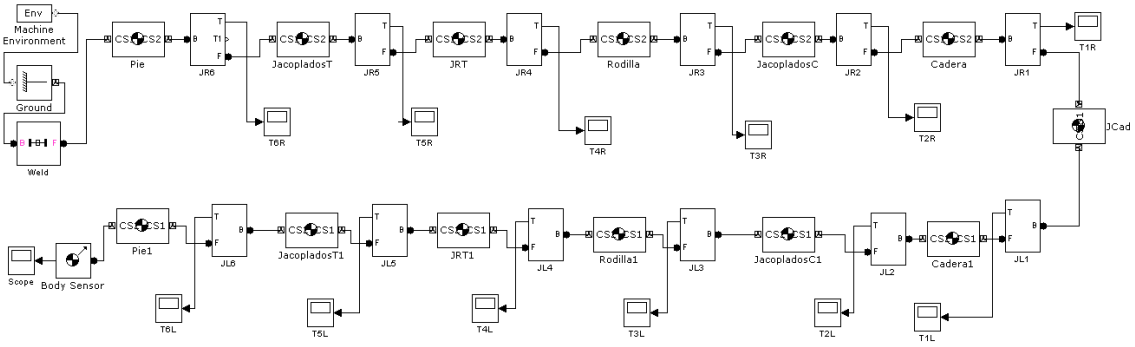


Figura 3.15: Diagrama de bloques de SimMechanics

Posteriormente se realiza un ensamble desde la base (pie fijo) hasta el efector final (pie flotante), de manera similar a la formulación Newton-Euler. Este ensamble se muestra en la Figura 3.15. Para conectar los eslabones se utilizan los bloques *Revolute*, los cuales corresponden a articulaciones rotacionales. Además, a este bloque se conectan los bloques *Joint Actuator* y *Joint Sensor*, los cuales se ocupan para introducir las trayectorias de referencia y para medir los pares ejercidos en las articulaciones respectivamente. El subsistema que conecta los eslabones se muestra en la Figura 3.16.

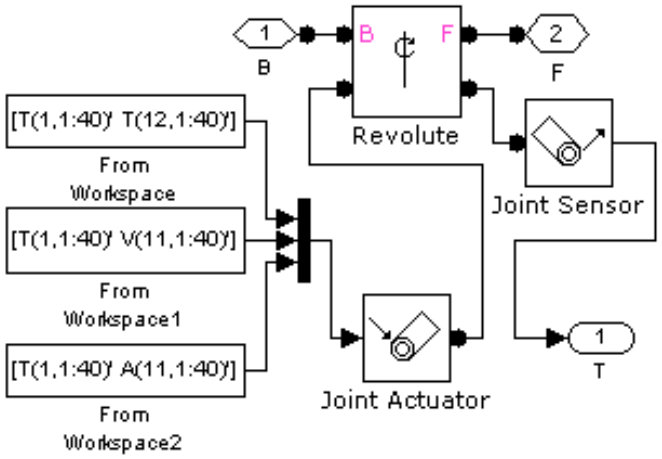
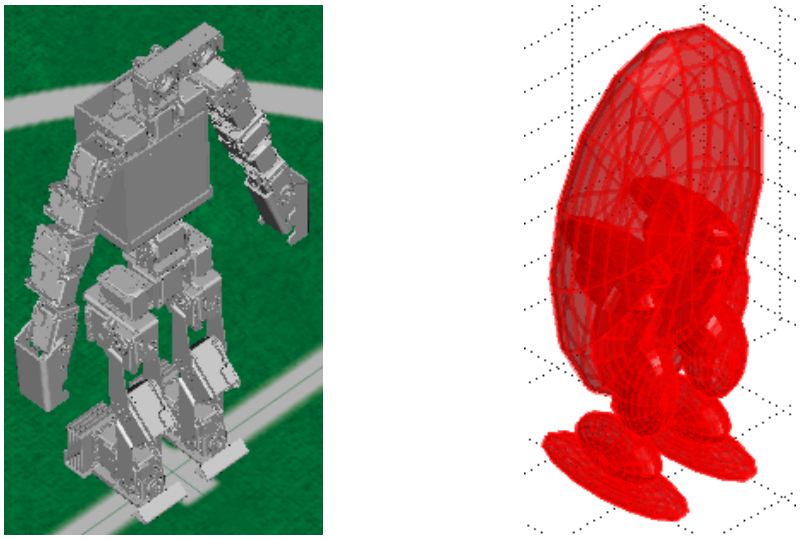


Figura 3.16: Subsistema de cada articulación

Una vez realizado el ensamble, se despliega el robot como un conjunto de elipsoides que representan la distribución de masa de cada eslabón. La configuración de los elipsoides nos sirve para corroborar que el modelo se ensambló de manera correcta, ya que el conjunto de elipsoides asemeja la estructura del robot, tal como se muestra en la Figura 3.17.



(a) Modelo VRML

(b) Modelo SimMechanics

Figura 3.17: Comparación de modelos ensamblados

Capítulo 4

Caminado Bípedo

Cuando caminamos, utilizamos una secuencia repetitiva de movimientos de las extremidades para mover el cuerpo hacia adelante mientras se mantiene simultáneamente la estabilidad de la postura. Mientras el cuerpo se mueve hacia adelante, una extremidad sirve como soporte mientras que la otra extremidad avanza hacia el nuevo sitio de soporte. Después, las extremidades invierten papeles. En esta fase, ambos pies se encuentran en contacto con el suelo y se realiza la transferencia del peso corporal de una extremidad a la otra. A la secuencia de estas funciones realizadas por una extremidad, se le llama *ciclo de marcha* [19]. La marcha representa la manera en que caminamos o corremos, por lo tanto cualquier caminado se realiza mediante cierta marcha. [20]

4.1. Conceptos

En esta parte se definirán algunos conceptos necesarios para el análisis del caminado, los cuales serán utilizados más adelante en los algoritmos implementados.

4.1.1. Fases del caminado

Cada ciclo de marcha se divide en dos periodos:

- Soporte. Es el término utilizado para designar el periodo completo en el cual el pie está en el suelo. Cuando ambos pies se encuentran en el suelo se le denomina *soporte doble*, de lo contrario *soporte sencillo*.

- Balanceo. El periodo de balanceo es el tiempo en el que el pie se encuentra en el

aire para el avance de la extremidad.

Frecuentemente estos periodos se denominan *fases de la marcha*. [19]

4.1.2. Planos de movimiento

Para describir las trayectorias de caminado en el espacio, utilizamos planos de movimiento. Existen tres planos de movimiento: el frontal, el sagital y el transversal. Estos planos se pueden observar en la Figura 4.1 La generación de trayectorias se basa principalmente en el movimiento en los planos frontal y sagital.

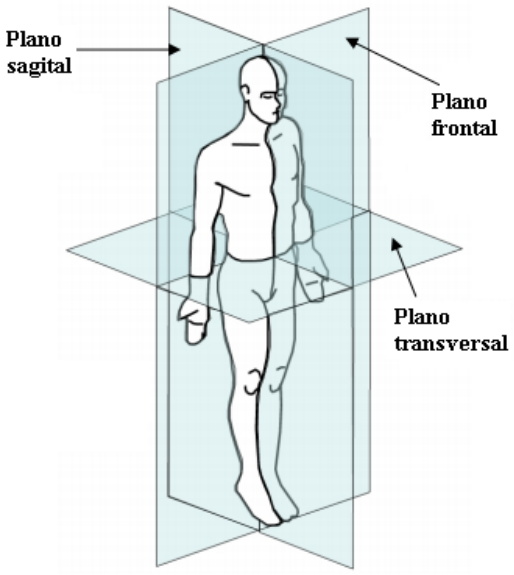


Figura 4.1: Planos de movimiento [21]

Estos planos son generados por ejes de movimiento que se tomarán como referencia para describir el movimiento de una articulación que posee más de un gdl. Para esto se utiliza la convención de ángulos *Roll*, *Pitch* y *Yaw*. Donde cada ángulo representa un movimiento angular alrededor de un eje perpendicular al plano frontal (eje sagital), sagital (eje frontal) y transversal (eje longitudinal), respectivamente.

4.2. Caminado Parametrizado

El caminado parametrizado fue un método trabajado por Oscar Vele [22] y posteriormente retomado por Rafael Cisneros [17]. Este método consiste en describir la trayectoria de caminado en dos planos: frontal y sagital. Para generar patrones de caminado, se toma en cuenta la trayectoria de la cadera y las trayectorias de los pies. Los parámetros que intervienen el plano sagital se muestran en la Figura 4.2,

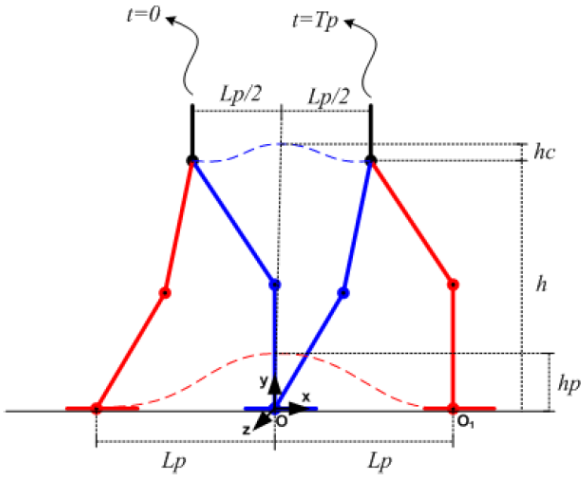


Figura 4.2: Parámetros del caminado en el plano sagital [22]

en donde:

- Tp = tiempo empleado por cada paso.
- Lp = longitud de paso.
- h = altura de la cadera.
- hc = rizado de la cadera.
- hp = altura máxima de la pierna flotante. [22]

Los parámetros utilizados en el plano frontal pueden observarse en la Figura 4.3,

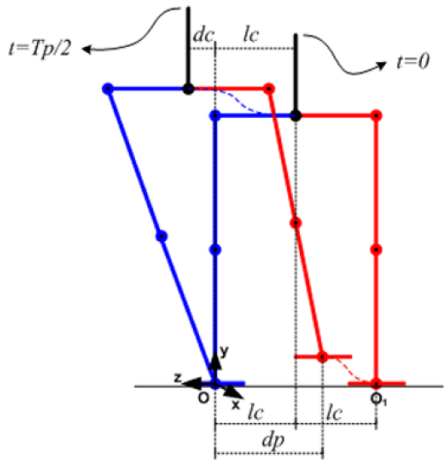


Figura 4.3: Parámetros del caminado en el plano frontal [22]

donde:

- lc = longitud del centro a uno de los extremos de la cadera.
- dc = sobrepaso del punto central de la cadera sobre la articulación del tobillo.
- dp = distancia entre pies a la mitad del paso.

Con estos parámetros, se obtiene las posiciones en el tiempo de la cadera y del pie flotante con respecto al pie de apoyo. Ya que estos parámetros no se encuentran definidos para todo el ciclo de caminado sino para algún tiempo determinado, las posiciones que faltan se obtienen por medio de interpolación segmentaria cúbica entre puntos donde la velocidad es igual a cero, de acuerdo a la siguiente ecuación

$$x(t) = x_0 + v_0 t + [\frac{3(x_1 - x_0) - T_i(2v_0 + v_1)}{T_i^2}]t^2 + [\frac{2(x_0 - x_1) + T_i(v_0 + v_1)}{T_i^3}]t^3 \quad (4.1)$$

donde:

- T_i = Periodo de interpolación.
- x_0 = Posición inicial.
- x_1 = Posición final.
- v_0 = Velocidad inicial.

- v_1 = Velocidad final.

Las posiciones de la cadera se obtienen a partir de un referencial fijo en el pie y las posiciones del pie flotante se obtienen a partir del referencial de la cadera. Entonces, la posición del pie flotante con respecto al pie de soporte se calcula como

$$T_{supp}^{float} = (T_{0sl}^{6sl})^{-1} T_{0sl}^{0fl} T_{0fl}^{6fl} \tag{4.2}$$

. Una vez obtenidas las trayectorias del pie y de la cadera, se utiliza cinemática inversa para obtener trayectorias en el espacio articular para transmitirlas a los servomotores. Para asegurar que esta solución es única, el referencial asociado a cada parte debe tener un plano que sea siempre paralelo al suelo. Una vez terminado el paso, el pie de soporte se convierte en el pie flotante y el pie flotante a su vez se convierte en el nuevo pie de soporte. De igual manera, cambiamos el referencial de referencia para la trayectoria del siguiente paso. Cabe mencionar que, este caminado está diseñado para trayectorias unidireccionales pero que pueden ser calculadas en línea sin necesidad de ser obtenidas previamente y se pueden modificar los parámetros de caminado, por ejemplo, aumentando la longitud del paso o aumentando la altura del pie para evadir obstáculos.

4.3. Caminado Omnidireccional

Para implementar un caminado omnidireccional, se utilizó el método propuesto por Behnke [23]. Este método difiere del anterior en que no utiliza cinemática inversa para determinar las posiciones de la pierna, aunque su parametrización es más compleja. Sin embargo, la gran ventaja de este método sobre el anterior es que se puede cambiar la dirección de caminado únicamente modificando el vector de entrada, además de que la dirección de caminado puede modificarse en línea. Además, las trayectorias de este método se calculan por medio de curvas senoidales en vez de splines.

Este método toma como entrada un vector de velocidades, la longitud de la pierna extendida y la longitud mínima que tendrá durante el caminado. Los tres elementos principales para generar caminado omnidireccional son: desplazamiento lateral del centro de masa, acortamiento de la pierna flotante y movimiento de las piernas en la

dirección de caminado.

4.3.1. Interfaz de la pierna

La pierna completa puede posicionarse relativa al tórax utilizando la extensión de la pierna (la distancia desde la articulación de la cadera a la articulación del tobillo), el ángulo de la pierna, y el ángulo del pie. Sea $\theta_{Leg} = (\theta^r_{Leg}, \theta^p_{Leg}, \theta^y_{Leg})$, el conjunto de ángulos que definen el ángulo de la pierna y sea $(\theta^r_{Foot}, \theta^p_{Foot})$ el conjunto de ángulos que definen el ángulo del pie, así como se muestra en la Figura 4.4.

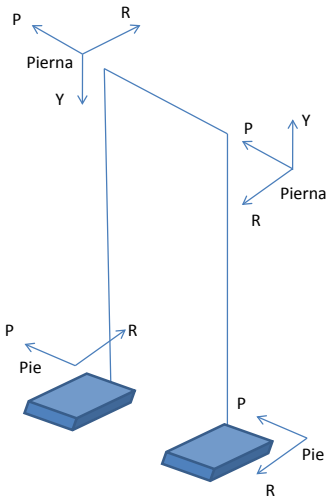


Figura 4.4: Ángulos de Roll, Pitch y Yaw de la pierna y del pie

Se define la extensión deseada de la pierna como $-1 \leq \gamma \leq 0$, cuando γ es igual a 0, la pierna se encuentra completamente extendida y cuando γ es igual a -1, la pierna se acorta a un porcentaje mínimo de su longitud original, definido por η_{min}. La longitud relativa deseada η puede calcularse como

$$\eta = 1 + (1 - \eta_{min})\gamma \tag{4.3}$$

Conociendo la longitud relativa podemos calcular el ángulo de la rodilla. como se muestra en la Figura 4.5.

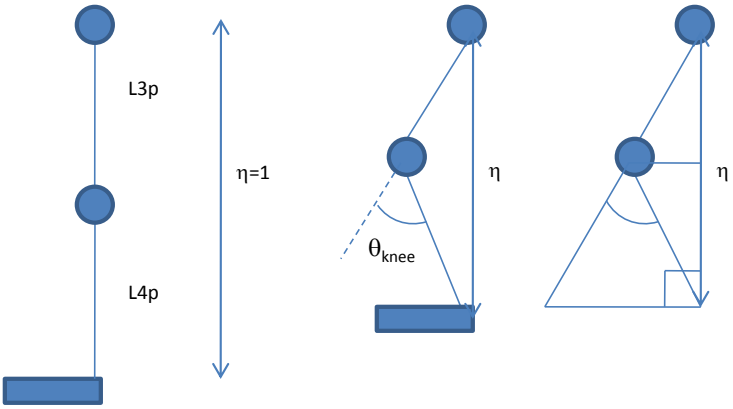


Figura 4.5: Análisis geométrico para obtener el ángulo de la rodilla

Utilizando los triángulos rectángulos, calculamos la variable articular

$$\theta_{Knee} = 180 - (90 - acos(\frac{\eta/2}{L_{4p}})) - (180 - 90 - acos(\frac{\eta/2}{L_{3p}})) \tag{4.4}$$

$$\theta_{Knee} = acos(\frac{\eta/2}{L_{4p}}) + acos(\frac{\eta/2}{L_{3p}}) = 2acos(\eta) \tag{4.5}$$

Se agrega un cambio de signo a la ecuación anterior de acuerdo al referencial asociado a la rodilla. Este ángulo acorta la pierna y se debe incluir en el ángulo del pie y de la pierna. Esto se hace restando el ángulo de la rodilla a los ángulos de *pitch* de la pierna y del pie. Si los eslabones tienen longitudes diferentes, se resta un porcentaje del ángulo de la rodilla, proporcional a la longitud que el eslabón de la articulación opuesta representa en la longitud total de la pierna. Ahora para el caso del pie, se debe restar también el ángulo de la pierna para compensar ambos cambios angulares. Estos nuevos valores angulares se actualizan como θ_{Hip} y θ_{Ankle}

$$\theta_{Hip} = \begin{pmatrix} \theta^r_{Leg} \\ \theta^p_{Leg} \end{pmatrix} + \begin{pmatrix} 0 \\ -L_{4p}\theta_{Knee} \end{pmatrix} \tag{4.6}$$

$$\theta_{Ankle} = \theta_{Foot} - \begin{pmatrix} \theta^r_{Leg} \\ \theta^p_{Leg} \end{pmatrix} + \begin{pmatrix} 0 \\ -L_{3p}\theta_{Knee} \end{pmatrix} \tag{4.7}$$

4.3.2. Reloj Central

Para generar las trayectorias de cada pierna, se utiliza una fase correspondiente a cada pierna, las cuales están desfasada $\pm\pi/2$ de una fase central ϕ_{Trunk} y se encuentran dentro del intervalo $[-\pi, \pi)$. Así como una frecuencia de paso ψ.

4.3.3. Vectores de entrada

Este algoritmo de caminado tiene como entrada un vector $\boldsymbol{V_{Robot}} = (v^x_{Robot}, v^y_{Robot}, v^\theta_{Robot})$ cuyos componentes son la velocidad en la dirección frontal, velocidad en la dirección lateral y velocidad de rotación alrededor del eje vertical.

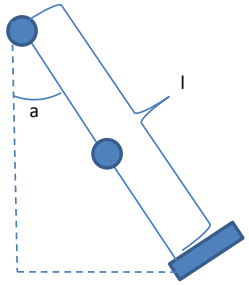


Figura 4.6: Modelo de péndulo para la pierna

El vector de amplitudes de balanceo $\boldsymbol{a_{Robot}} = (a^r_{Robot}, a^p_{Robot}, a^y_{Robot})$ se obtiene a partir del vector de velocidad. Considerando la pierna como un péndulo, como se muestra en la Figura 4.6, tenemos que las amplitudes son:

$$(a^r_{Robot}, a^p_{Robot}) = (asin(\frac{v^x_{Robot}}{\psi l}, asin(\frac{v^y_{Robot}}{\psi l}))) \tag{4.8}$$

$$a^y_{Robot} = \frac{v^\theta_{Robot}}{\psi} \tag{4.9}$$

Podemos notar que la longitud del paso está dada por $\frac{v_{Robot}}{\psi}$.

4.3.4. Desplazamiento

El desplazamiento lateral del centro de masa se realiza de una manera senoidal como sigue:

$$\theta_{Shift} = a_{Shift} sin(\phi_{Leg}) \tag{4.10}$$

donde a_{Shift} es la amplitud de desplazamiento, la cual se incrementa con la velocidad de la marcha, así como con la velocidad lateral. Este desplazamiento se logra mediante el uso de las variables angulares de *roll*. El uso de ambas variables permite desplazar el centro de masa, así como mantener erguida la unidad pasajera.

$$\theta_{LegShift} = \theta_{Shift} \tag{4.11}$$

$$\theta_{FootShift} = -0.5\,\theta_{Shift} \tag{4.12}$$

4.3.5. Acortamiento

Mientras se desplaza el robot lateralmente, la pierna contraria no es necesaria para soportar su peso y puede acortarse. El tiempo que dura esta fase, se determina por:

$$\phi_{Short} = v_{Short}(\phi_{Leg} + \pi/2 + o_{Short}) \tag{4.13}$$

donde v_{Short} determina la duración y o_{Short} determina el corrimiento de fase relativo al desplazamiento lateral del peso. Para realizar la transición entre la pierna extendida y la pierna acortada, se utiliza un coseno de la siguiente manera:

$$\gamma_{Short} = -0.5\,a_{Short}(cos(\phi_{Short}) + 1) \tag{4.14}$$

donde a_{Short} es la amplitud de acortamiento e incrementa con la velocidad de la marcha. Para levantar el pie al final, dirigido a la dirección de caminado, se utiliza:

$$\theta_{FootShort} = -0.125\,a^{p}_{Robot}(cos(\phi_{Short}) + 1) \tag{4.15}$$

Esto evita contacto accidental entre la parte frontal del pie y el piso, durante el balanceo.

4.3.6. Carga

Cuando la pierna de balanceo, toca el suelo, ésta es acortada por segunda vez con el propósito de facilitar la carga, de la siguiente manera:

$$\phi_{Load} = v_{Load}\, piCut(\phi_{Leg} + \pi/2 + -\pi/v_{Short} + o_{Short}) - \pi \tag{4.16}$$

donde v_{Load} determina la duración del segundo acortamiento y la función $piCut$ mapea su argumento al rango $[-\pi, \pi)$. El acortamiento se calcula nuevamente utilizando un coseno, de la siguiente manera:

$$\gamma_{Load} = -0.5\, a_{Load}(cos(\phi_{Load}) + 1) \tag{4.17}$$

donde la amplitud del segundo acortamiento a_{Load} depende de la amplitud de balanceo en la dirección frontal a^p_{Robot}.

4.3.7. Balanceo

Después de que la pierna ha sido descargada y acortada, se mueve rápidamente en la dirección de caminado y al mismo tiempo, la pierna de soporte completamente extendida, se mueve lentamente en la misma dirección.El balanceo se revierte lentamente durante el resto del ciclo de caminado. Este balanceo se calcula de la siguiente manera:

$$\phi_{Swing} = v_{Swing}(\phi_{Leg} + \pi/2 + o_{Swing}) \tag{4.18}$$

donde v_{Swing} es la velocidad de balanceo y o_{Swing} es el corrimiento de fase. Mientras el balanceo es sinusoidal, el movimiento se revierte linealmente.

$$\theta_{Swing} = \begin{cases} sin(\phi_{Swing}) & \text{si } -\pi/2 \leq \phi_{Swing} < \pi/2 \\ b(\phi_{Swing} - \pi/2) - 1 & \text{si } \pi/2 \leq \phi_{Swing} \\ b(\phi_{Swing} + \pi/2) - 1 & \text{de otra manera} \end{cases} \tag{4.19}$$

donde b es la velocidad del movimiento en reversa. El balanceo se hace con el ángulo de la pierna y se compensa parcialmente con el ángulo del pie, como se muestra a

continuación:

$$\theta^r_{LegSwing} = ls\, a^r_{Robot}\theta_{Swing} \tag{4.20}$$

$$\theta^p_{LegSwing} = a^p_{Robot}\theta_{Swing} \tag{4.21}$$

$$\theta^y_{LegSwing} = ls\, a^y_{Robot}\theta_{Swing} \tag{4.22}$$

$$\theta^r_{FootSwing} = 0.25\, a^r_{Robot}\theta_{Swing} \tag{4.23}$$

$$\theta^p_{FootSwing} = 0.25\, ls\, a^p_{Robot}\theta_{Swing} \tag{4.24}$$

donde ls es el signo de la pierna: $ls = -1$ para la pierna izquierda $ls = 1$ y para la pierna derecha.

4.3.8. Equilibrio

El robot mantiene su equilibrio al inclinarse con cada paso, en el plano sagital y el plano lateral, así como compensando los ángulos de la pierna y del pie. El ángulo de *roll* de la pierna asegura que las piernas no colisionen mientras el robot camina lateralmente o mientras rota.

4.3.9. Resultado

Los componentes individuales del movimiento de caminado se combinan de la siguiente manera:

$$\theta^r_{Leg} = \theta^r_{LegSwing} + \theta_{LegShift} + \theta^r_{LegBal} \tag{4.25}$$

$$\theta^p_{Leg} = \theta^p_{LegSwing} \tag{4.26}$$

$$\theta^y_{Leg} = \theta^y_{LegSwing} \tag{4.27}$$

$$\theta^r_{Foot} = \theta^r_{FootSwing} + \theta_{FootShift} + \theta^r_{FootBal} \tag{4.28}$$

$$\theta^p_{Foot} = \theta^p_{FootSwing} + \theta_{FootShort} + \theta^p_{FootBal} \tag{4.29}$$

$$\gamma = \gamma_{Short} + \gamma_{Load} \tag{4.30}$$

4.4. Estabilidad del caminado

Un aspecto de mucha importancia es evitar la caída del humanoide mientras se encuentra caminando. Para prevenir caídas, una condición necesaria y suficiente es asegurar que el contacto entre el pie y el suelo en cada instante sea una superficie y no una línea o un punto, esto se traduce en mantener su balance, tanto dinámico como estático. Cabe aclarar que el término balance se utiliza aquí en el sentido de mantener una posición erguida del humanoide en general. Tomando en cuenta que el humanoide se encuentra bajo la influencia de fuerzas inerciales, cuya dirección e intensidad cambian durante el caminado, el balance dinámico adquiere mayor importancia. Se dice que la marcha es balanceada dinámicamente si no existe rotación del pie de soporte alrededor de su borde durante el caminado.

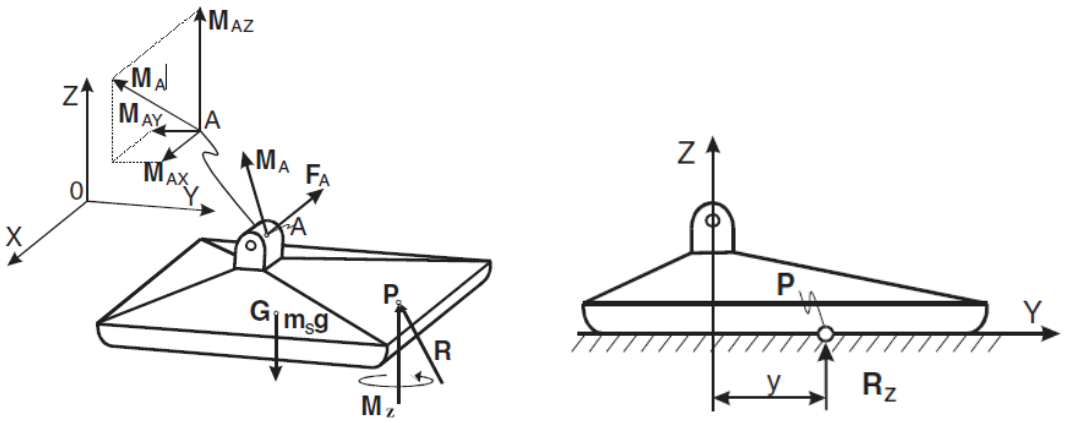
Para que exista balance dinámico, es necesario y suficiente que la resultante de las fuerzas normales de reacción, actúen en un punto que se encuentra dentro del área de soporte, la cual es la superficie determinada por el contacto entre el pie y el suelo. En la fase de soporte simple, el área de soporte coincide con la envolvente convexa determinada por el pie en contacto con el suelo, mientras que en fase de soporte doble, el área de soporte es la envolvente convexa determinada por las áreas de los pies y el piso, y sus tangentes comunes, tales que el área envuelta sea máxima. Este concepto también se conoce como *polígono de soporte*.

Debido al carácter unilateral del contacto mecánico entre el pie y el suelo, las fuerzas normales de reacción, se encuentran en una dirección (desde el suelo hacia el pie). Si hallamos un punto en donde sólo actúe la resultante de las fuerzas normales, la ausencia de cualquier par significa que los momentos serán iguales a cero para cualquier eje que pasa por este punto y además es tangencial al suelo. A este punto se le conoce como punto de momento cero o *Zero-Moment Point* (ZMP) [20].

4.4.1. Derivación del Zero-Moment Point

Considerando el robot en su fase de soporte simple, analizamos las fuerzas que actúan sobre el pie. Primero descartamos la parte del robot por encima del tobillo y

reemplazamos su influencia por una fuerza $\boldsymbol{F}_A$ y un momento $\boldsymbol{M}_A$, como se muestra en la Figura 4.7a , mientras que el peso del pie actúa en su centro de gravedad (G). El pie también experimenta la reacción del suelo en el punto P, cuya acción mantiene al robot en equilibrio.



(a) Fuerzas sobre el pie ejercidas por el cuerpo del robot y el suelo

(b) Momento M_x compensado al desplazar punto P en el plano $y - z$

Figura 4.7: Fuerzas que actúan sobre el pie del robot. [24]

En general, la reacción total del suelo consiste en tres componentes de la fuerza $\boldsymbol{R}(R_x, R_y, R_z)$ y del momento $\boldsymbol{M}(M_x, M_y, M_z)$. Suponiendo que no existe deslizamiento en el contacto entre el pie y el suelo, la fricción estática compensará los componentes horizontales de la fuerza (R_x, R_y) y el momento de reacción vertical (M_z). Debido a la naturaleza unidireccional del contacto entre el pie y el suelo, los componentes horizontales de los momentos pueden compensarse únicamente cambiando la posición de la fuerza de reacción $\boldsymbol{R}$ dentro del polígono de soporte. Por lo tanto, las componentes horizontales del momento $\boldsymbol{M_A}$ desplazarán la fuerza de reacción a una posición donde se equilibre la carga adicional. En el caso planar mostrado en la Figura 4.7b, el momento M_{Ax} es equilibrado al desplazar el punto donde actúa la fuerza R_z, una distancia y. Cabe destacar que, mientras la fuerza de reacción se encuentre dentro del área de soporte, el incremento en el momento del tobillo será compensado al cambiar la posición de esta fuerza, y se anularán los momentos M_x y M_y. Sin embargo, si el incremento es significativo, la fuerza actuará en el borde del pie y existirá una parte sin compensar

del momento de reacción, la cual causará que el robot gire alrededor del borde del pie y probablemente caiga. Por lo tanto, podemos decir que la condición necesaria y suficiente para que el robot se encuentre en equilibrio dinámico es que, para el punto P en la planta, donde la fuerza de reacción del piso actúa, los momentos horizontales M_x y M_y sean iguales a cero. Dada la dinámica del robot de la Figura 4.7a, podemos calcular la posición del ZMP que asegura equilibrio dinámico partiendo de las ecuaciones de equilibrio estático para el pie de soporte como sigue :

$$\boldsymbol{R} + \boldsymbol{F}_A + m_s g = 0 \tag{4.31}$$

$$\overrightarrow{\mathbf{OP}} \times \overrightarrow{\mathbf{R}} + \overrightarrow{\mathbf{OG}} \times m_s g + \boldsymbol{M}_A + M_z + \overrightarrow{\mathbf{OA}} \times \boldsymbol{F}_A = 0 \tag{4.32}$$

donde $\overrightarrow{\mathbf{OP}}$, $\overrightarrow{\mathbf{OG}}$ y $\overrightarrow{\mathbf{OA}}$ son vectores que van del origen del sistema de coordenadas O_{xyz} al punto de reacción del suelo (P), centro de masa del pie (G), y articulación del tobillo (A), respectivamente, mientras que m_s es la masa del pie. En el caso general, M_z es diferente de cero y sólo puede reducirse a través de la dinámica del robot. Sin embargo, la proyección de la Ecuación 4.32 en el plano horizontal nos da la ecuación:

$$(\overrightarrow{\mathbf{OP}} \times \overrightarrow{\mathbf{R}})^H + \overrightarrow{\mathbf{OG}} \times m_s g + \boldsymbol{M}_A^H + (\overrightarrow{\mathbf{OA}}\ times \boldsymbol{F}_A)^H = 0, \tag{4.33}$$

la cual es la base para calcular la posición de la fuerza de reacción del suelo que actúa en el punto P. [24]

4.4.2. Cálculo del ZMP

Para un movimiento dado del robot, la fuerza y el momento en la articulación del tobillo ($\boldsymbol{F}_A$ y $\boldsymbol{M}_A$) pueden obtenerse a partir del modelo dinámico del robot, y todos los elementos de la ecuación Ecuación 4.33 con excepción de $\overrightarrow{\mathbf{OP}}$ serán conocidos. El procedimiento para determinar la posición del ZMP consiste en 2 pasos:

1. Calcular $\overrightarrow{\mathbf{OP}}$ de la Ecuación 4.33. Llamemos la posisición obtenida del punto P *posición calculada del ZMP* ya que en este momento no sabemos si esta posición se encuentra dentro del polígono de soporte o afuera.

2. Se compara la posición calculada del ZMP con el tamaño del polígono de soporte.

Si el ZMP calculado se encuentra fuera del polígono de soporte, esto significa que la fuerza de reacción del suelo, actuando en el punto P, se encuentra en el borde del polígono de soporte y se iniciará la rotación del robot alrededor del borde pie debido al momento desequilibrado, cuya intensidad depende de la distancia del polígono de soporte a la posición calculada del ZMP. Si el punto calculado donde actúa la fuerza de reacción del suelo se encuentra dentro del polígono de soporte, este punto es el ZMP y el robot se encuentra en equilibrio. [24]

4.4.3. Modelo simplificado

Dadas las trayectorias de movimiento del robot y su modelo dinámico, podemos calcular o predecir el ZMP resultante. Esto depende del cálculo del modelo dinámico del robot completo, lo cual tiene un costo computacional. Para evitarnos esto, podemos reducir el robot completo a un mecanismo más sencillo, esto es, un modelo carro-mesa, como el mostrado en la Figura 4.8.

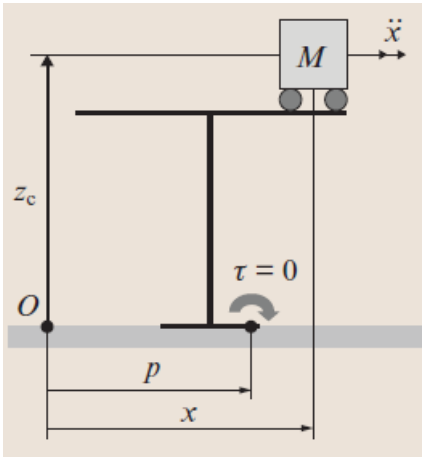


Figura 4.8: Modelo dinámico simplificado para el cálculo del ZMP. [2]

Este modelo consiste en un carro que se desplaza sobre una mesa sin masa. El carro tiene masa M y su posición (x, z_c) corresponde al centro de masa del robot. También, la mesa se considera que tiene el mismo polígono de soporte que el robot. El par τ alrededor del punto p está dado por

$$\tau = -Mg(x - p) + M\ddot{x}z_c \tag{4.34}$$

donde g es la aceleración debida a la gravedad. Utilizando la condición de momento cero de $\tau = 0$, el ZMP calculado de este modelo carro-mesa se obtiene como

$$p = x - \frac{z_c}{g}\ddot{x}. \tag{4.35}$$

Podemos notar que cuando la aceleración de este carro es cero, el ZMP corresponde a la proyección del Centro de Masa ($p = x$) [2].

Capítulo 5

Sistema de Visión

Un sistema de visión es fundamental para un robot móvil. La información obtenida le permite conocer el entorno donde se encuentra y así puede decidir cómo interactuar con él. Esta información le permite planear trayectorias para desplazarse y realizar la tarea para la cual fue programado. En el caso del fútbol, es necesario poder desplazarse hacia donde se encuentra la pelota, tomando en cuenta su ubicación, así como la de los demás robots que se encuentran en el campo y además la portería hacia la cual se deberá dirigir. Cuando además el robot se programa para manipular objetos, no sólo debe ser capaz de reconocer los objetos que manipulará, así como su pose, también debe calcular la distancia a la que estos se encuentran. Esto se puede hacer de 2 maneras: utilizando marcas conocidas o utilizando visión estereoscópica. El sistema de percepción del prototipo AH1N2 se basa en visión estereoscópica obtenida mediante la cámara Minoru y OpenCV.

5.1. Visión Estereoscópica

La visión estereoscópica es la habilidad de obtener información de una escena, a partir de dos o más imágenes tomadas desde distintos puntos de vista. Nuestra percepción 3D del mundo se debe a la interpretación que el cerebro da a partir de la diferencia en la posición retinal, llamada *disparidad* , entre puntos correspondientes. Las disparidades de todos los puntos de una imagen forman el mapa de disparidad y si se conoce la geometría del sistema esteroscópico, el mapa puede convertirse en un mapa 3D de la escena observada (reconstrucción). [25]

En la práctica, visión estereoscópica involucra cuatro pasos:

1. Matemáticamente remover distorsión radial y tangencial. Los resultados de este paso son imágenes sin distorsión.

2. Ajuste para los ángulos y las distancias entre cámaras. Los resultados de este paso son imágenes cuyas filas están alineadas y rectificadas.

3. Encontrar las mismas características en las imágenes izquierda y derecha (correspondencia). El resultado de este paso es un mapa de disparidad, donde las disparidades son las diferencias en coordenadas x, en los planos imagen, de la misma característica observada en la cámara izquierda y en la derecha.

4. Conociendo la geometría del sistema estereoscópico, transformamos el mapa de disparidad en distancias por medio de triangulación. [26]

5.1.1. Calibración de cámaras

La calibración de cámaras es un proceso que se utiliza para corregir las desviaciones causadas por los lentes de las cámaras mediante el modelo geométrico de la cámara y el modelo de distorsión de los lentes. Ambos modelos definen los *parámetros intrínsecos* de la cámara y nos permiten relacionar las mediciones de la cámara con mediciones en el mundo tridimensional. Ya que se utiliza una cámara estereoscópica, el proceso de calibración no sólo calcula los parámetros intrínsecos de cada cámara, también calcula las relaciones geométricas entre las 2 cámaras, esto es, calcula la matriz de rotación y el vector de traslación entre las dos cámaras.

Una vez obtenidos los parámetros de calibración, se obtienen los mapas de consulta para realizar la rectificación de imágenes. Estos mapas indican de dónde se deben interpolar píxeles de la imagen de origen para cada píxel de la imagen de destino. Posteriormente se utilizan estos mapas para corregir la distorsión en las imágenes. La rectificación estéreo es el proceso de corregir las imágenes individuales para que aparezcan como si hubiesen sido tomadas por dos cámaras con planos imagen cuyas filas están alineadas. En este caso, utilizamos el algoritmo de Bouguet para la rectificación

estéreo. Dadas las matrices de rotación y traslación (R,T) entre las imágenes estéreo, el algoritmo de Bouguet minimiza el cambio que la reproyección genera para cada una de las dos imágenes (minimizando las distorsiones) mientras que maximiza el área de visión común. El resultado de este algoritmo son dos matrices de rectificación para alinear filas, dos matrices para las ecuaciones de proyección y un parámetro opcional Q, que es la matriz de reproyección. Este último parámetro es el que se utiliza para calcular la distancia en tres dimensiones.

5.1.2. Cálculo de distancia 3D

Para calcular la distancia 3D de cualquier punto en la escena, es necesario conocer el modelo geométrico del sistema estereoscópico. En este caso partimos de un sistema cuyas cámaras se encuentran paralelas y cuyos planos imagen son coplanares, como se muestra en la Figura 5.1. En este sistema, los píxeles en ambas imágenes tienen sus orígenes en la parte superior izquierda de sus respectivas imágenes, y los píxeles se denotan por coordenadas (x_l,y_l) y (x_r,y_r), respectivamente. Los centros de proyección están en O_l y O_r, con rayos principales intersecando el plano imagen en el punto principal (c_x,c_y). Tomando x^l y x^r como las posiciones horizontales de puntos en las imágenes izquierda y derecha respectivamente, la disparidad se define simplemente como $d = x^l - x^r$.

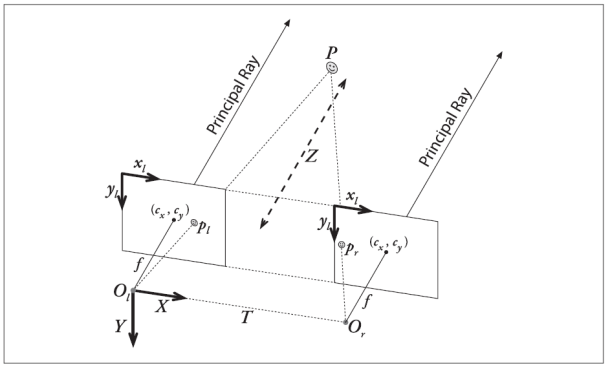


Figura 5.1: Sistema estereoscópico [26]

Puntos en dos dimensiones pueden transformarse a puntos en tres dimensiones, dadas las coordenadas en pantalla y la matriz de parámetros intrínsecos de la cámaras. Para esto utilizamos la matriz de reproyección Q obtenida en la calibración estéreo. La

matriz de reproyección es:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & (c_x - c'_x)/T_x \end{bmatrix} = \begin{bmatrix} q00 & q01 & q02 & q03 \\ q10 & q11 & q12 & q13 \\ q20 & q21 & q22 & q23 \\ q30 & q31 & q32 & q33 \end{bmatrix}$$

Aquí los parámetros pertenecen a la imagen izquierda excepto c'_x, que es la coordenada x del punto principal en la imagen derecha. Dado un punto homogéneo de dos dimensiones y su disparidad asociada d, podemos reproyectar el punto a tres dimensiones utilizando:

$$Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

Las coordenadas 3D entonces son (X/W, Y/W, Z/W)

5.2. OpenCV

5.2.1. Calibración de Cámaras

Para calibrar la cámara estereoscópica se siguen los siguientes pasos:

1. Se imprime un tablero como el que se muestra en la Figura 5.2.

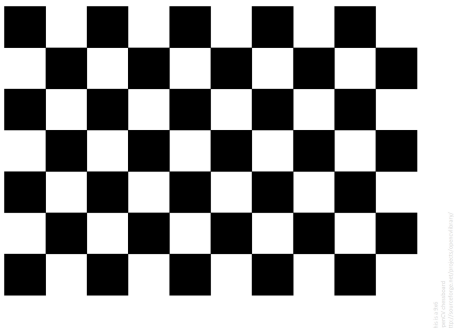


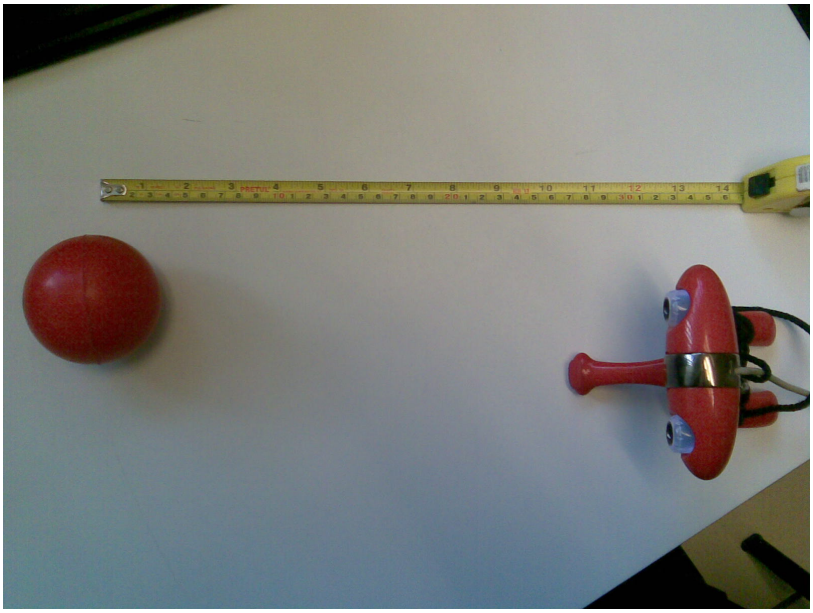
Figura 5.2: Tablero para calibración [27]

2. Utilizando el código publicado en [28], se toma una secuencia de fotos al tablero en diversas posiciones, utilizando ambas cámaras. Además, se crea un archivo de texto con la lista de imágenes, que se utiliza en la calibración.

3. Ahora se ejecuta el código de calibración publicado en [27], utilizando como parámetros de entrada el nombre del archivo de texto, el número de cuadros horizontales y verticales que detectará el algoritmo y por último la longitud del lado de los cuadrados que componen el tablero.

   ./stereo_calibrate list.txt 9 6 2.6

   El código almacena en diferentes archivos las matrices obtenidas durante la calibración. En este caso las matrices que nos interesan son las matrices Q, *mx1*, *mx2*, *my1* y *my2*, las cuales son la matriz de reproyección y los mapas de consulta en las coordenadas x y y.



(a) Escena

(b) Medición real

Figura 5.3: Configuración de la escena para medición 3D

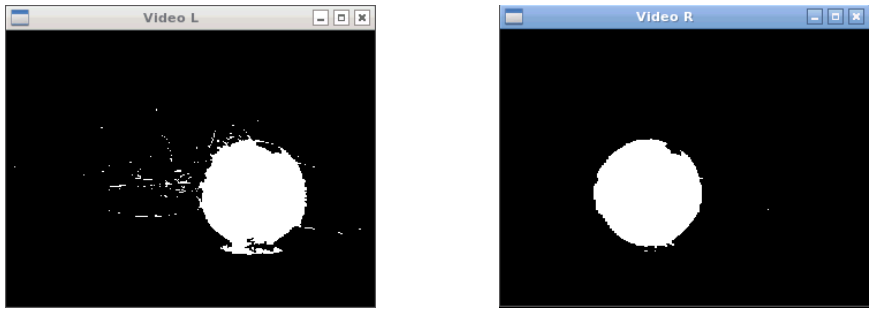
5.2.2. Distancia 3D

Una vez obtenida la calibración de cámaras ahora sólo resta utilizar las matrices obtenidas en conjunto con imágenes rectificadas para determinar distancia en tres dimensiones. Para esto se siguen los siguientes pasos:

1. Se toma una imagen, con ambas cámaras, del objeto de interés. En este caso, el objeto con el cual se desea estimar distancia en tres dimensiones, es una pelota roja. La escena se puede observar en la Figura 5.3.

2. Se utilizan las matrices que representan a los mapas de consulta para realizar una rectificación de las imágenes utilizando la función *cvRemap*.

3. Se segmentan las imágenes para encontrar un punto en común en ambas. En este caso se utilizó únicamente segmentación por color en el espacio HSV. Para esto se utilizó la función *cvThreshold* con las imágenes de *Hue* y *Saturation*. Posteriormente se utilizó una operación binaria AND y la función *cvFloodFill* para reducir puntos segmentados fuera de la pelota. El resultado de esta calibración se puede observar en la Figura 5.4.



(a) Imagen Izquierda (b) Imagen Derecha

Figura 5.4: Imágenes segmentadas

4. Una vez segmentada la pelota, se obtiene su centroide utilizando momentos de OpenCV. Estos momentos se utilizan para realizar una comparación de contornos de una forma sencilla mediante las funciones *cvMoments* y *cvGetSpatialMoments*.

5. Utilizando el centroide de ambas imágenes, se realiza la operación matricial con el vector de coordenadas del centroide en la imagen izquierda, la disparidad y la matriz Q para obtener la distancia de la pelota con respecto a la cámara izquierda expresada en centímetros.

$$X = xl * q00 + q03$$
$$Y = yl * q11 + q13$$
$$Z = q23$$
$$W = d * q32 + q33$$

donde qij son los elementos de la matriz Q, xl y yl son las coordenadas x y y, respectivamente, del centroide en la imagen izquierda y d es la disparidad. Posteriormente se divide X, Y y Z entre W para obtener la distancia 3D. La distancia calculada se puede observar en la Figura 5.5.

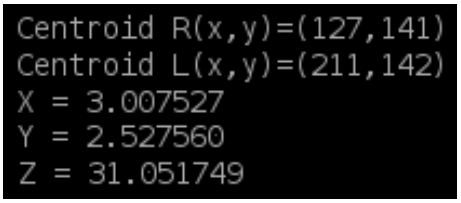


Figura 5.5: Distancia calculada a partir de dos imágenes

El código se encuentra en el Apéndice G.3.

5.3. Coordinación Percepción-Control

En esta sección se explica la manera en que la distancia obtenida en la sección anterior, se relaciona con la cinemática inversa del brazo para tomar objetos que estén al alcance del robot.

5.3.1. Cinemática de la cámara

Ya que obtuvimos las coordenadas de un objeto con respecto a la cámara, ahora necesitamos transformar las coordenadas al referencial del pecho para posteriormente transformarlo al referencial del hombro y aplicar la cinemática inversa del brazo con el cual se desee alcanzar dicho objeto. Para esto se necesita un nuevo diagrama cinemático que incluya el referencial del pecho y el referencial de la cámara, de acuerdo a la metodología de DH, el cual se muestra en la Figura 5.6. El referencial base *b1* se

encuentra en el centro del servomotor del pecho y el referencial C se encuentra en el centro de la cámara estereoscópica. Una vez establecidos los referenciales, se obtienen los parámetros de DH mostrados en la Tabla 5.1

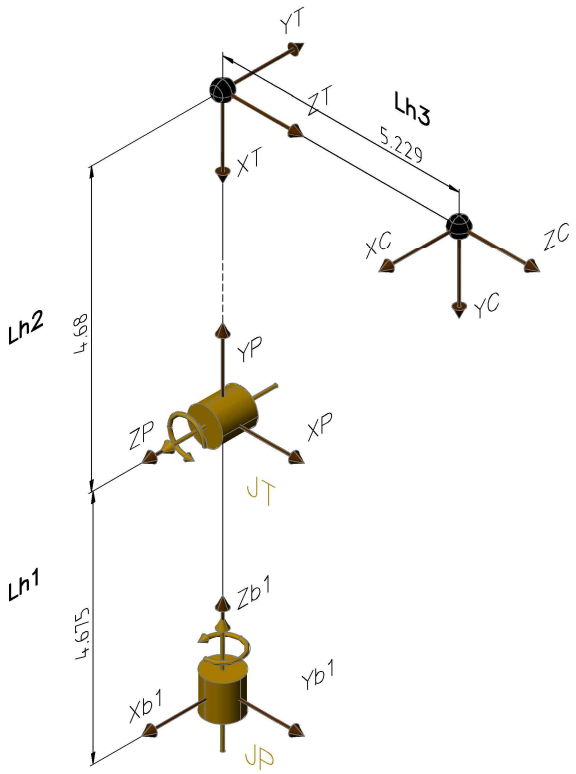


Figura 5.6: Diagrama cinemático de la cabeza

articulación i	θ_i	α_i	a_i	d_i
1	$\theta_P + 90°$	90°	0	L_{h1}
2	θ_T - 90°	-90°	$-L_{h2}$	0
3	$-90°$	0°	0	L_{h2}

Tabla 5.1: Tabla de parámetros de la cabeza

Utilizando la Tabla 5.1 obtenemos las matrices de paso A_i

$$A_1 = \begin{bmatrix} -s_P & 0 & c_P & 0 \\ c_P & 0 & s_P & 0 \\ 0 & 1 & 0 & L_{h1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} s_T & 0 & c_T & -L_{h2}s_T \\ -c_T & 0 & s_T & L_{h2}c_T \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & L_{h3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para obtener la transformación homogénea del referencial de la cámara con respecto al del pecho realizamos la multiplicación de matrices

$$A_1A_2A_3 = T_{b1}^C = \begin{bmatrix} c_P & -s_Ps_T & -s_Pc_T & -L_{h3}s_Pc_T + L_{h2}s_Ps_T \\ s_P & c_Ps_T & c_Pc_T & L_{h3}c_Pc_T - L_{h2}c_Ps_T \\ 0 & -c_T & s_T & L_{h3}s_T + L_{h2}c_T + L_{h1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

y las coordenadas de la cámara con respecto al referencial del pecho quedan de la siguiente manera:

$$\begin{bmatrix} X_{b1} \\ Y_{b1} \\ Z_{b1} \end{bmatrix} = \begin{matrix} X_C\ c_P + (L_{h2} - Y_C)\ s_Ps_T - (Z_C + L_{h3})\ s_Pc_T \\ X_C\ s_P + (Y_C - L_{h2})\ c_Ps_T + (Z_C + L_{h3})\ c_Pc_T \\ (L_{h2} - Y_C)\ c_T + (Z_C + L_{h3})\ s_T + L_{h1} \end{matrix} \tag{5.2}$$

5.3.2. Detección de objetos

Para el algoritmo de visión en conjunto con el movimiento del brazo, se utilizó un proceso de detección más riguroso ya que se debía verificar que los ángulos fuesen adecuados para el brazo antes de transmitirlos a los servomotores. Para esta nueva detección se realizó un proceso de segmentación por color, similar al mencionado en la Sección 5.2.2 pero también se utilizó la transformada de Hough para detección de círculos [29].

5.3.3. Seguimiento y alcance de objetos

Una vez detectada la pelota, se realizó un seguimiento de la misma y posteriormente el robot decide en qué momento tratar de agarrarla, esto es, en el momento en que estuviera al alcance de su brazo. Para realizar el seguimiento de la pelota, se consideró la posición del centroide dentro de un intervalo donde los ángulos de *Pan* y de *Tilt* no serían modificados, reduciendo el número de ángulos que se enviarían a los servomotores. Además, la detección de círculos, a pesar de haber sido mejorada, presenta pequeñas variaciones en la detección de la pelota y, por lo tanto, de su centroide. Cabe aclarar que el intervalo considerado se aplica directamente en los ángulos de *Pan* y de *Tilt*, esto es, primero se calculan estos ángulos utilizando las coordenadas de la pelota con respecto a la cámara de la manera siguiente:

$$Pan = atan2(\frac{-X}{Z}) \quad Tilt = atan2(\frac{-Y}{Z})$$

y si alguno de ellos es mayor que el intervalo establecido, se actualiza la posición de los servomotores utilizando este nuevo ángulo para centrar la pelota nuevamente. Una vez obtenida la posición de la pelota con respecto a la cámara, utilizamos los ángulos de *Pan* y de *Tilt* para transformar las coordenadas al referencial del pecho mediante la cinemática obtenida en la sección previa. Ahora verificamos si esta posición se encuentra dentro del espacio de trabajo del robot utilizando la longitud del brazo y además debemos verificar que los ángulos obtenidos no sobrepasen los límites articulares establecidos. Si se determina que la pelota se encuentra dentro del espacio de trabajo y además los ángulos son adecuados, se manda la instrucción a los servomotores para que intente alcanzar y agarrar la pelota.

Capítulo 6

Pruebas y Resultados

En este capítulo se mostrarán los resultados obtenidos de la implementación de los algoritmos desarrollados, tanto en simulación como en el prototipo físico. Estos resultados ayudarán a comprobar la validez de dichos algoritmos para que el trabajo futuro sobre dicho prototipo pueda ser desarrollado en simulación y posteriormente implementado en el prototipo sin la necesidad de realizar mayores cambios.

6.1. Cinemática

Para las pruebas de cinemática, se especificaron diferentes poses utilizando cinemática directa dentro del simulador. Posteriormente las poses calculadas se utilizaron como entrada para el simulador de cinemática inversa y también para los programas en la Roboard.

6.1.1. Piernas

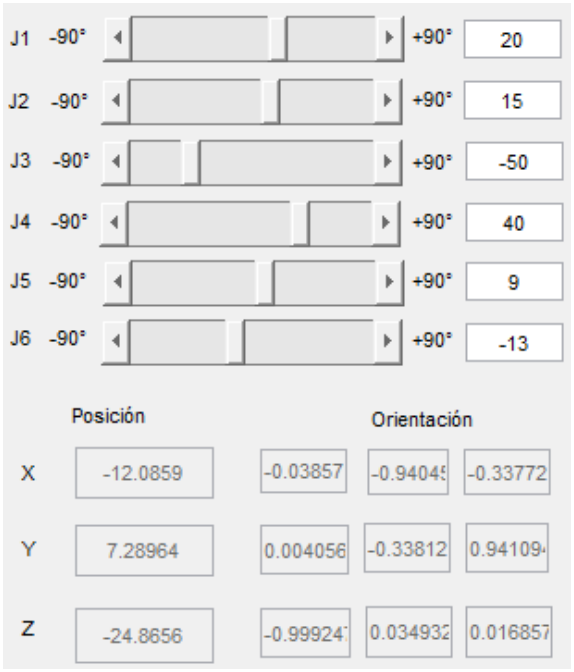
Como primera prueba se especificó el conjunto de ángulos mostrado en la Figura 6.1a. Este conjunto posiciona al pie izquierdo adelante y distante al eje longitudinal, flexionando la rodilla y además orienta al pie casi paralelo al suelo. Esta pose se muestra

en la Figura 6.1b. Esta pose está dada por la matriz de transformación

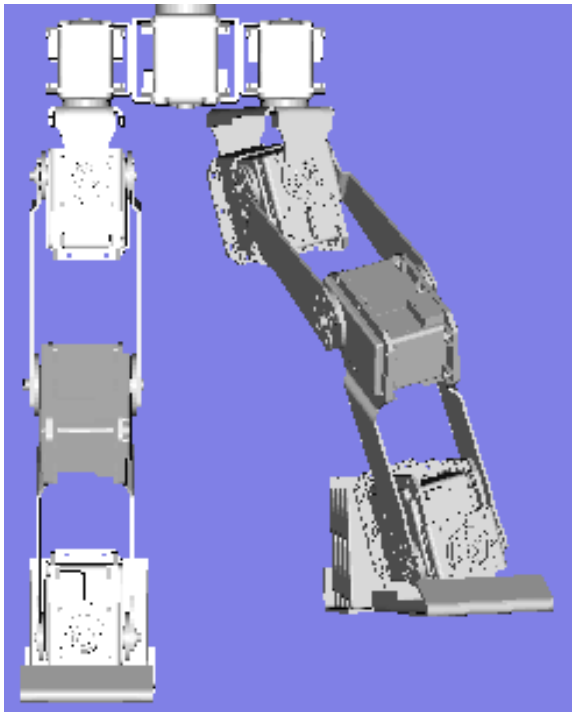
$$T_{b2}^{6} = \begin{bmatrix} -0.03857 & -0.94045 & -0.33772 & -12.0859 \\ 0.004056 & -0.33812 & 0.941094 & 7.28964 \\ -0.999242 & 0.034932 & 0.016857 & -24.8656 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

la cual se introdujo al simulador de cinemática inversa y al calcular el conjunto de ángulos que generaban dicha pose, se obtuvieron los resultados mostrados en la Figura 6.2.

Estos ángulos posicionaron el modelo virtual (Figura 6.3a) y el prototipo físico (Figura 6.3b) utilizando como entrada, la pose obtenida mediante la cinemática directa.

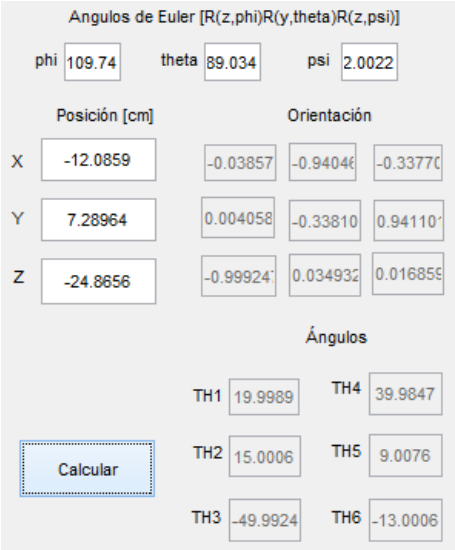


(a) Conjunto de ángulos de la pierna y su respectiva pose obtenida



(b) Pose obtenida mediante el conjunto de ángulos de la pierna

Figura 6.1: Cinemática directa de la pierna en la prueba uno

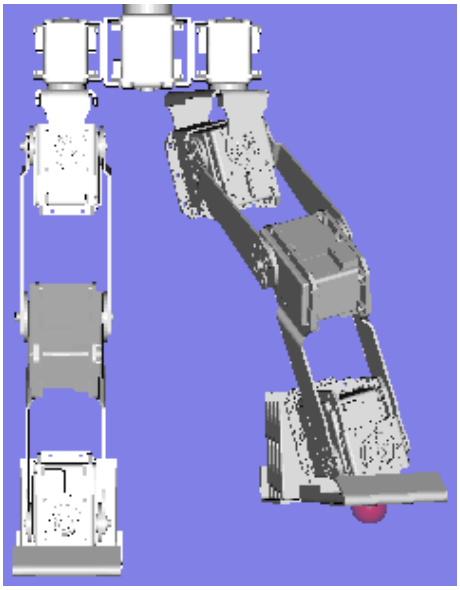


(a) Pose deseada para el modelo VRML y sus variables articulares resultantes

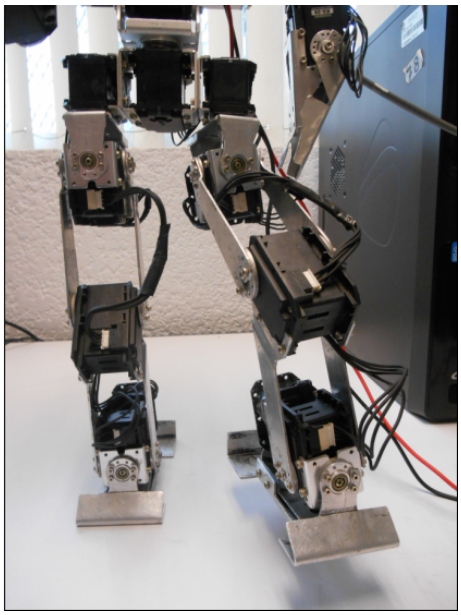
```
Pose deseada =
-0.038570 -0.940450 -0.337720 -12.085900
0.004056 -0.338120 0.941094 7.289640
-0.999242 0.034932 0.016857 -24.865600
0.000000 0.000000 0.000000 1.000000

 Variables articulares obtenidas
THL1 = 20.000087
THL2 = 14.999897
THL3 = -49.994090
THL4 = 39.987095
THL5 = 9.007033
THL6 = -13.000004
THR1 = 1.100000
THR2 = 1.100000
THR3 = 1.100000
THR4 = 1.100000
THR5 = 1.100000
THR6 = 1.100000
```

(b) Pose deseada para el prototipo y sus variables articulares resultantes

Figura 6.2: Cinemática inversa de la pierna en la prueba uno



(a) Pose alcanzada por el modelo VRML de la pierna

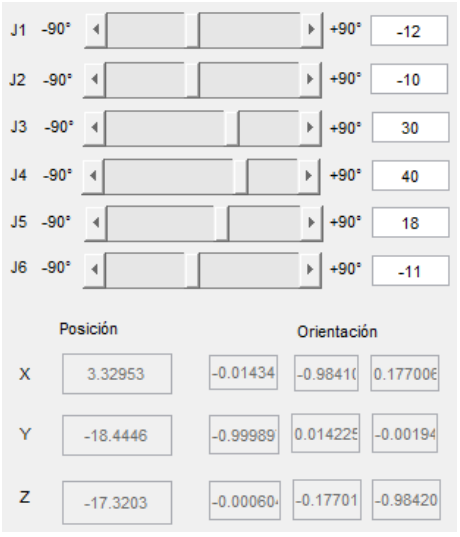


(b) Pose alcanzada por el prototipo físico de la pierna

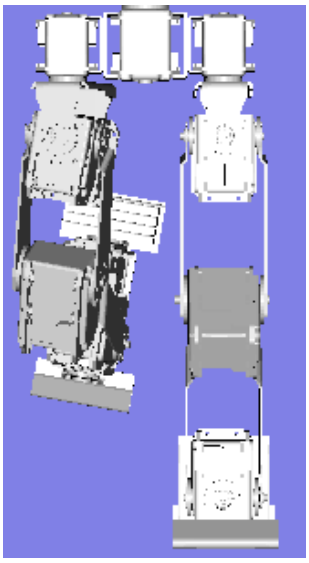
Figura 6.3: Poses obtenidas mediante cinemática inversa de la pierna en la prueba uno

Para la segunda prueba se especificó el conjunto de ángulos mostrado en la Figura 6.4a. Este conjunto posiciona al pie derecho atrás y cercano al eje longitudinal,

flexionando la rodilla y además orientado con la suela casi perpendicular al suelo. Esta pose se muestra en la Figura 6.4b.

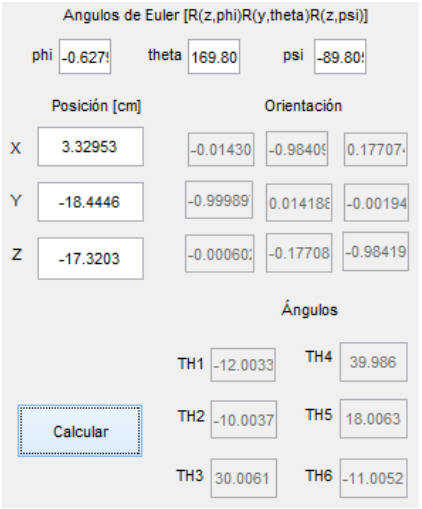


(a) Conjunto de ángulos de la pierna y su respectiva pose obtenida

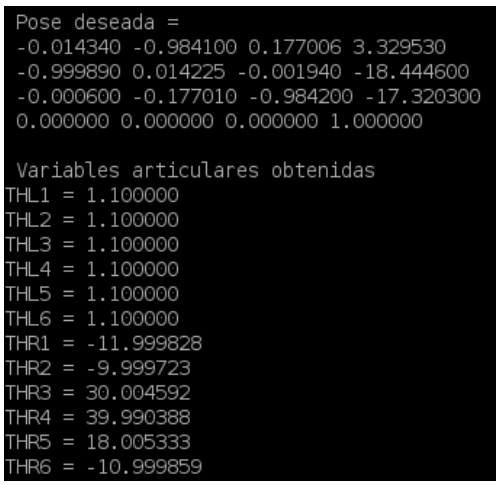


(b) Pose obtenida mediante el conjunto de ángulos de la pierna

Figura 6.4: Cinemática directa de la pierna en la prueba dos



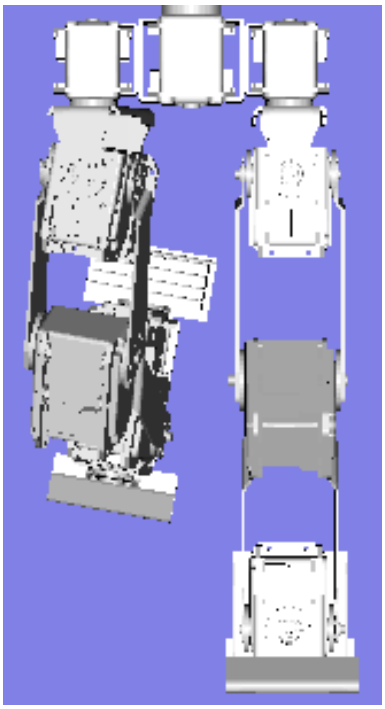
(a) Pose deseada para el modelo VRML y sus variables articulares resultantes



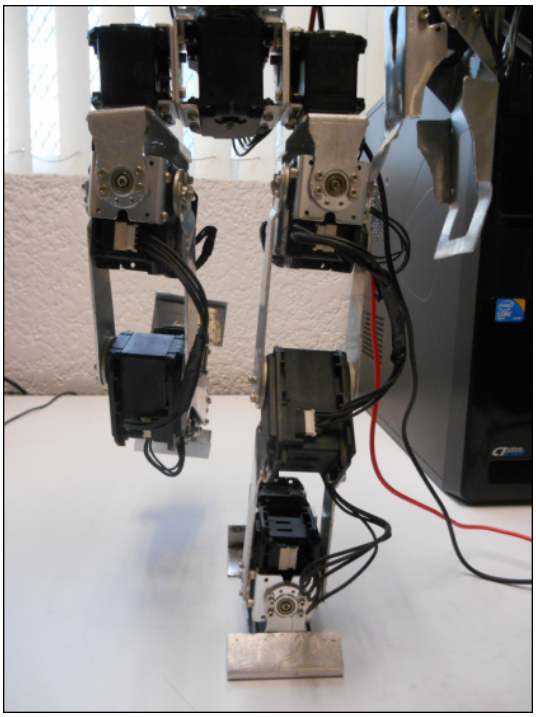
(b) Pose deseada para el prototipo y sus variables articulares resultantes

Figura 6.5: Cinemática inversa de la pierna en la prueba dos

Estos ángulos posicionaron nuevamente el modelo virtual como se puede observar en la Figura 6.6a y el prototipo físico en la Figura 6.6b.



(a) Pose alcanzada por el modelo VRML de la pierna



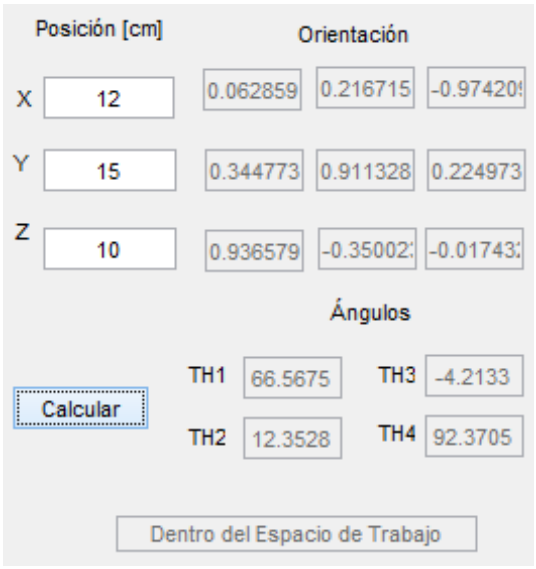
(b) Pose alcanzada por el prototipo físico de la pierna

Figura 6.6: Poses obtenidas mediante cinemática inversa de la pierna en la prueba dos

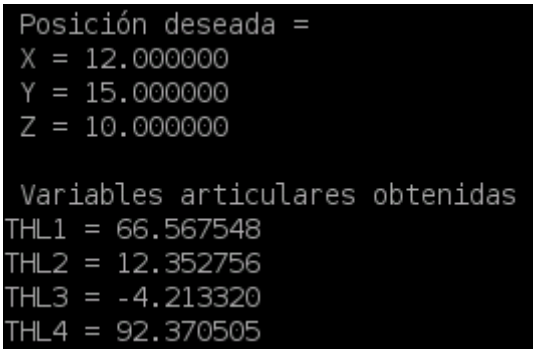
6.1.2. Brazos

Para las pruebas de los brazos se especificó únicamente la posición deseada para la mano y posteriormente se comparó con la posición alcanzada mediante la cinemática inversa. Ambas posiciones se pueden observar dentro del simulador utilizando una pelota para la posición deseada y otra para la posición de la mano. Se realizaron cuatro pruebas para comparar la precisión de la cinemática. Estas pruebas se realizaron en los 4 cuadrantes del espacio de trabajo del brazo, determinados por la intersección de los ejes de *Pitch* y *Yaw* del hombro. Finalmente se muestran diversas tablas de comparación donde se observa el error numérico expresado en centímetros.

- Prueba uno.

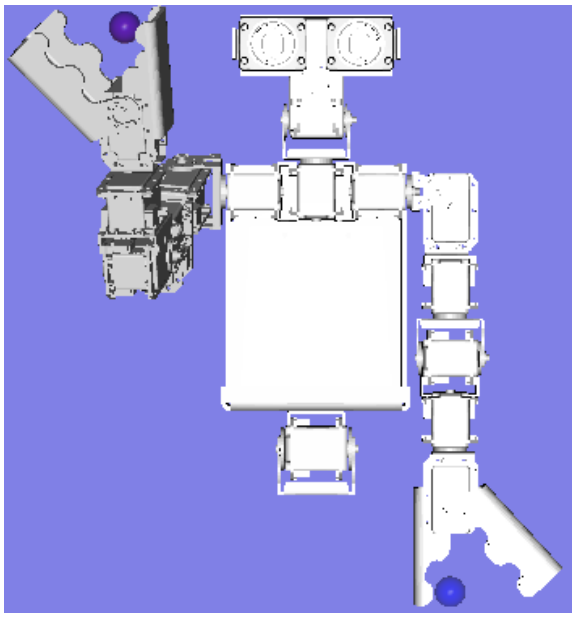


(a) Posición deseada para el modelo VRML y sus variables articulares resultantes

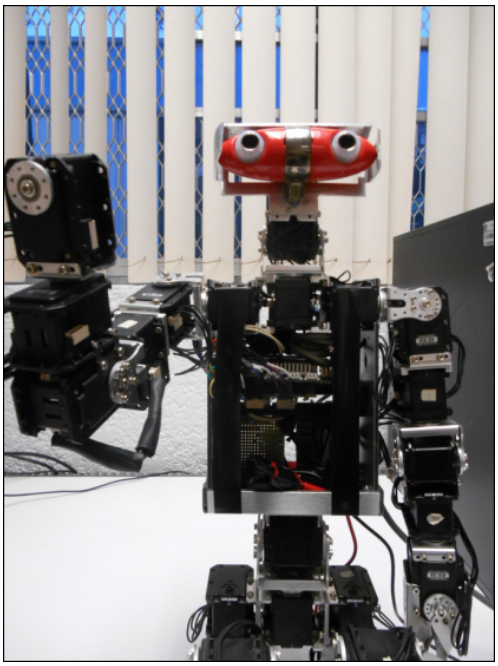


(b) Posición deseada para el prototipo y sus variables articulares resultantes

Figura 6.7: Cinemática inversa del brazo en la prueba uno



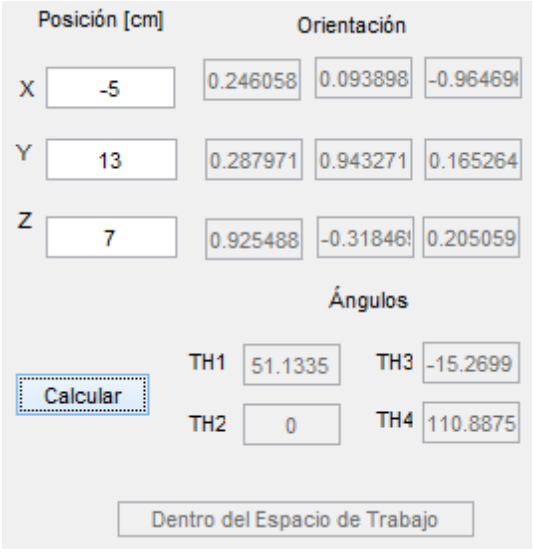
(a) Posición alcanzada por el modelo VRML



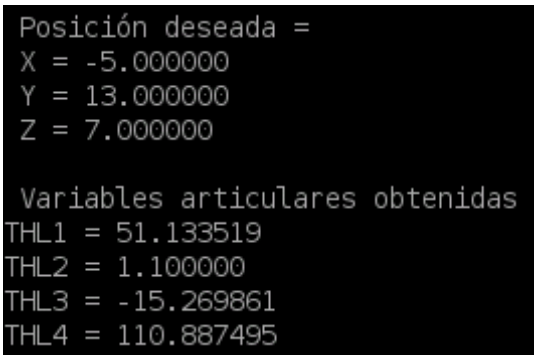
(b) Posición alcanzada por prototipo físico

Figura 6.8: Poses obtenidas mediante cinemática inversa del brazo en la prueba uno

- Prueba dos.

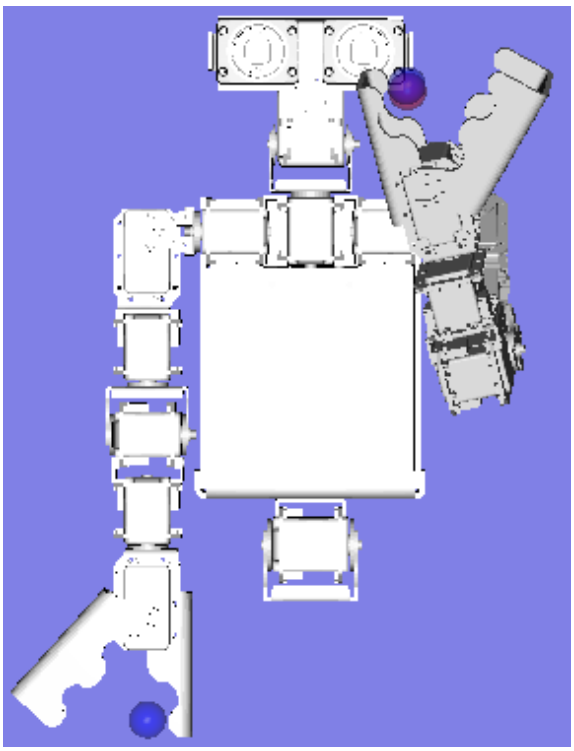


(a) Posición deseada para el modelo VRML y sus variables articulares resultantes

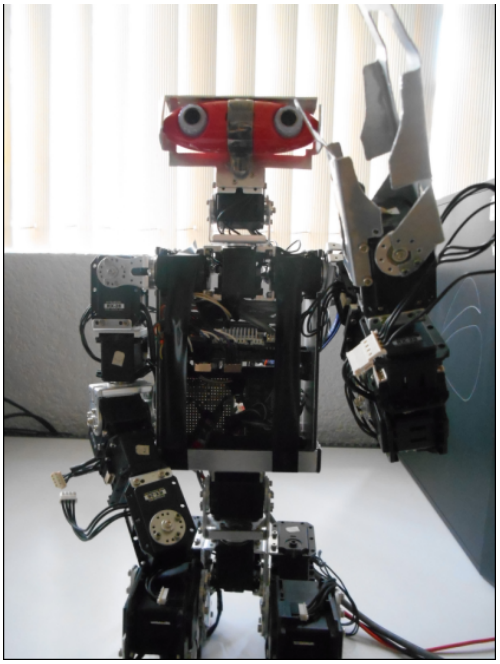


(b) Posición deseada para el prototipo y sus variables articulares resultantes

Figura 6.9: Cinemática inversa del brazo en la prueba dos



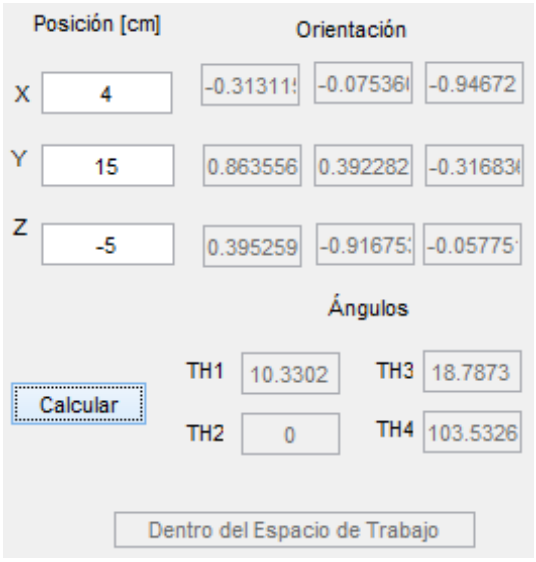
(a) Posición alcanzada por el modelo VRML



(b) Posición alcanzada por prototipo físico

Figura 6.10: Poses obtenidas mediante cinemática inversa del brazo en la prueba dos

- Prueba tres.

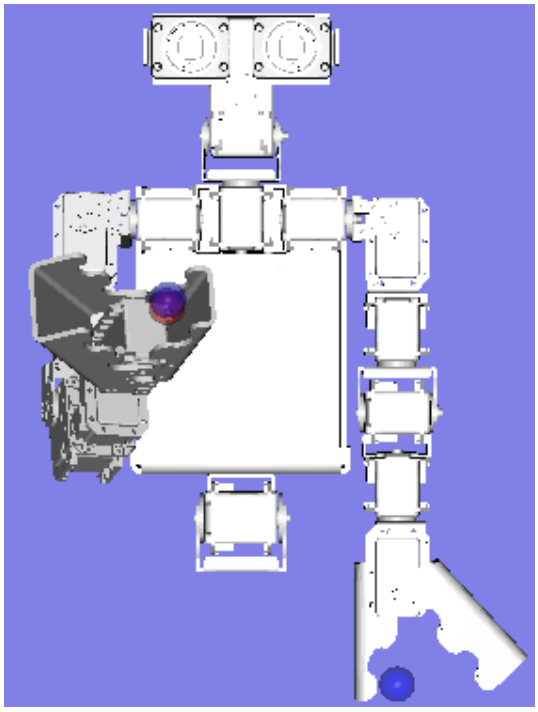


(a) Posición deseada para el modelo VRML y sus variables articulares resultantes

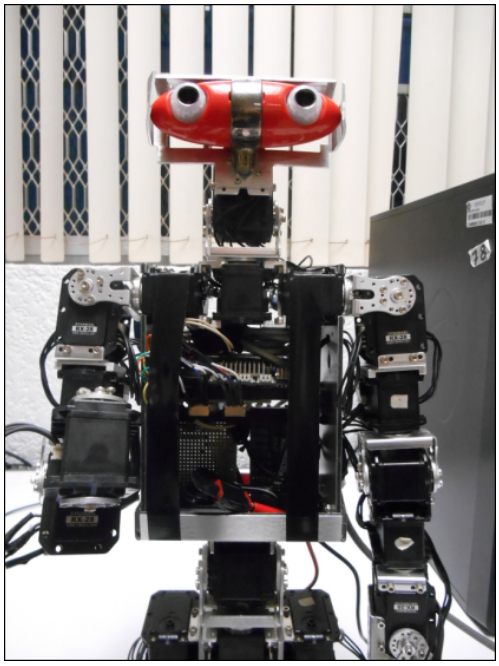
```
Posición deseada =
X = 4.000000
Y = 15.000000
Z = -5.000000

Variables articulares obtenidas
THL1 = 10.330156
THL2 = 1.100000
THL3 = 18.787337
THL4 = 103.532618
```

(b) Posición deseada para el prototipo y sus variables articulares resultantes

Figura 6.11: Cinemática inversa del brazo en la prueba tres



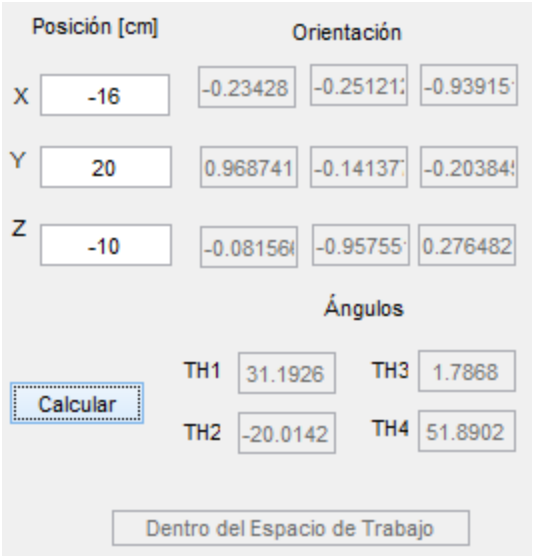
(a) Posición alcanzada por el modelo VRML



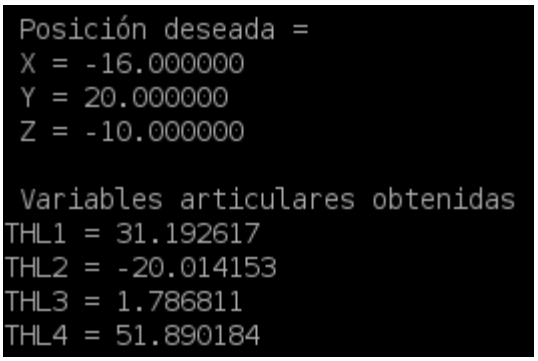
(b) Posición alcanzada por prototipo físico

Figura 6.12: Poses obtenidas mediante cinemática inversa del brazo en la prueba tres

- Prueba cuatro.

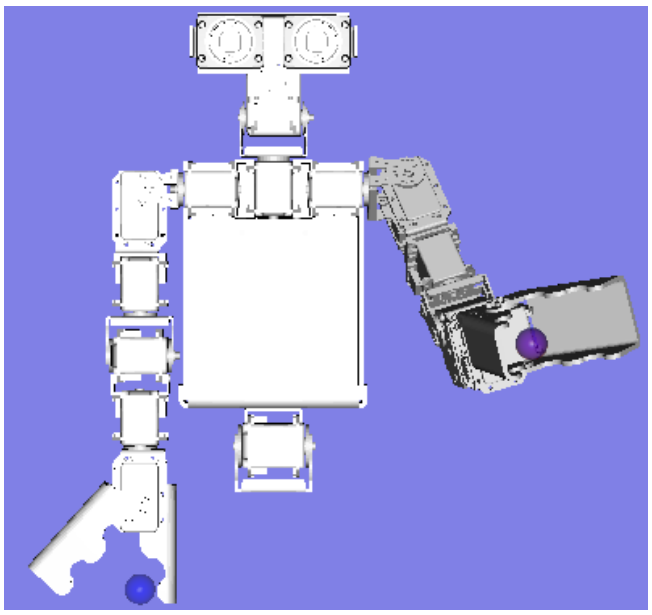


(a) Posición deseada para el modelo VRML y sus variables articulares resultantes

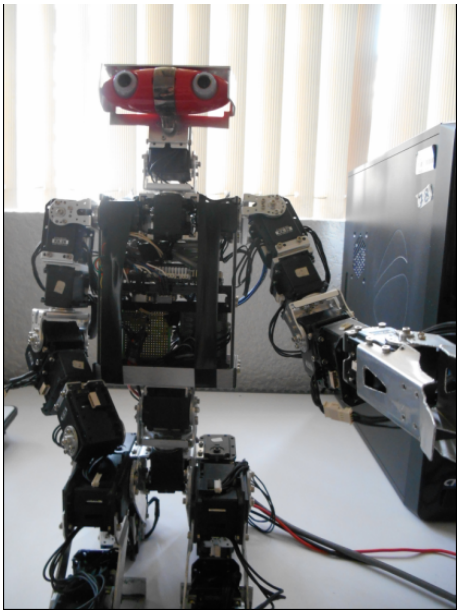


(b) Posición deseada para el prototipo y sus variables articulares resultantes

Figura 6.13: Cinemática inversa del brazo en la prueba cuatro



(a) Posición alcanzada por el modelo VRML



(b) Posición alcanzada por prototipo físico

Figura 6.14: Poses obtenidas mediante cinemática inversa del brazo en la prueba cuatro

P. deseada	P. real	Error
12	11.9888	0.0112
15	14.9567	0.0433
10	10.0684	0.0684
Magnitud del Error		0.0817

Tabla 6.1: Tabla de comparación de la cinemática inversa del brazo en la prueba uno

P. deseada	P. real	Error
-5	-4.9651	0.0349
13	12.8171	0.1829
7	7.3117	0.3117
Magnitud del Error		0.363

Tabla 6.2: Tabla de comparación de la cinemática inversa del brazo en la prueba dos

P. deseada	P. real	Error
5	4.9702	0.0298
15	15.0766	0.0766
-5	-4.7406	0.2594
Magnitud del Error		0.2721

Tabla 6.3: Tabla de comparación de la cinemática inversa del brazo en la prueba tres

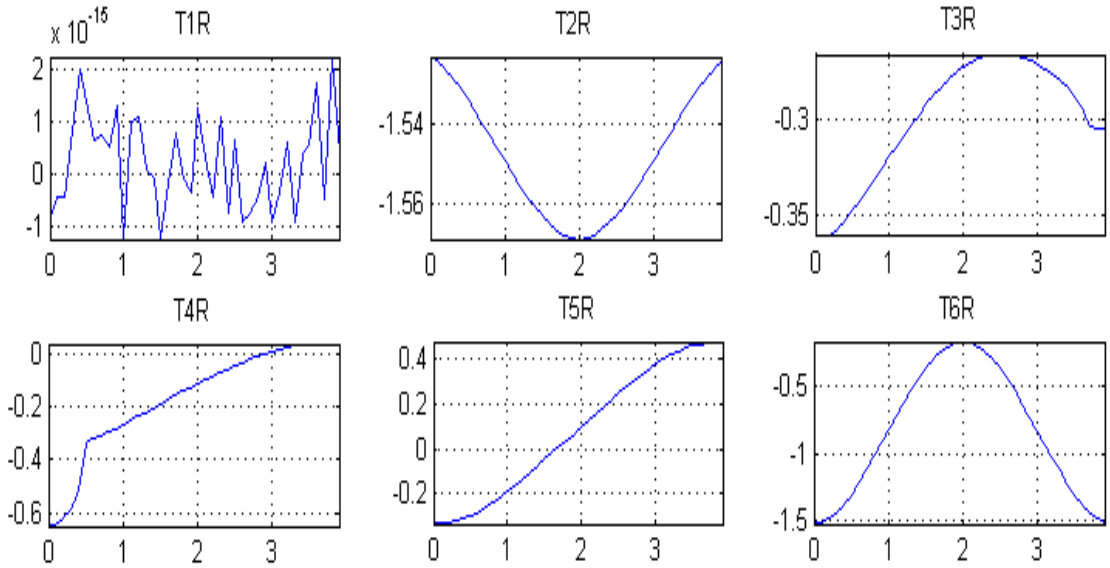
P. deseada	P. real	Error
-16	-16.9903	0.0097
20	20.0315	0.0315
-10	-9.9439	0.0561
Magnitud del Error		0.0651

Tabla 6.4: Tabla de comparación de la cinemática inversa del brazo en la prueba cuatro

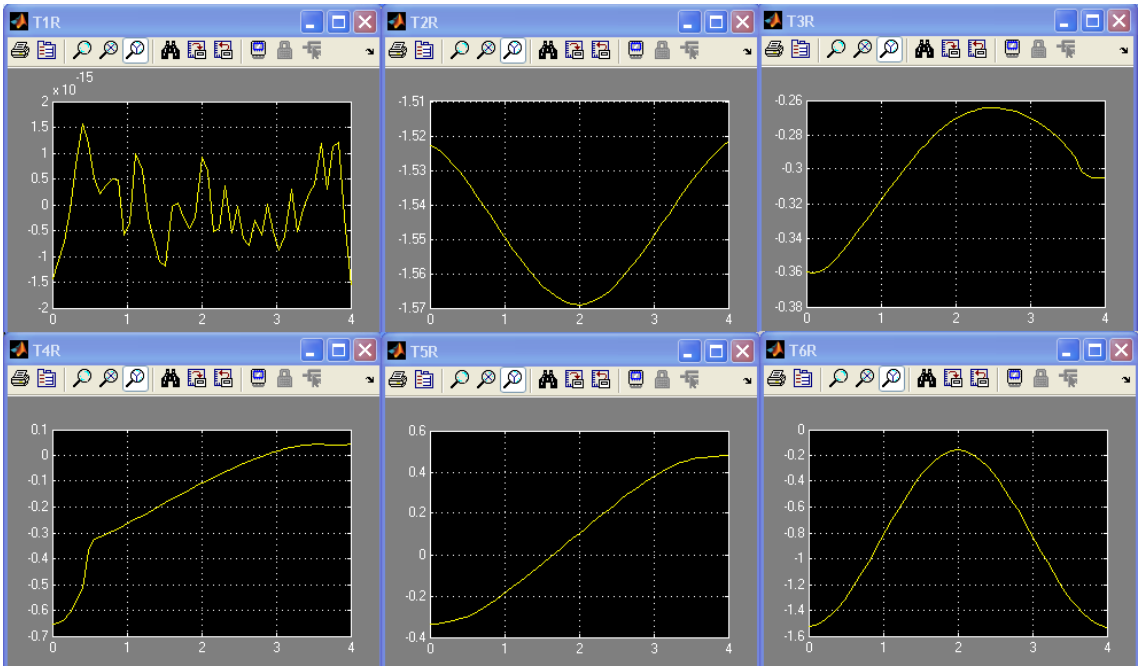
6.2. Dinámica

Para comprobar la validez del modelo dinámico obtenido mediante la formulación de Newton-Euler, se realizó el ensamble del robot en SimMechanics como se mencionó en la Sección 3.3.2. La trayectoria que se evaluó en esta parte, fue simplemente un paso desde la posición inicial del caminado parametrizado. Al introducir la posición, velocidad y aceleración, junto con el tiempo de muestreo, a los modelos, obtenemos los

pares resultantes que será necesario aplicar en cada articulación con el fin de producir dicho movimiento. Los pares resultantes en la pierna derecha (soporte), se muestran en la Figura 6.15. Los pares resultantes en la pierna izquierda (flotante), se muestran en la Figura 6.16.

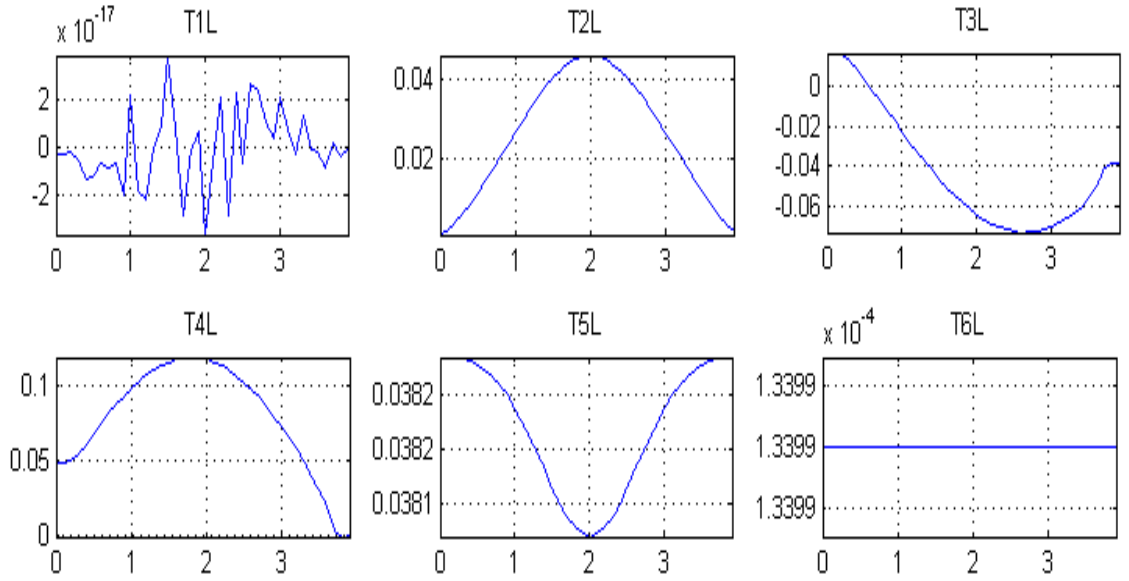


(a) Pares obtenidos con el modelo de Newton-Euler

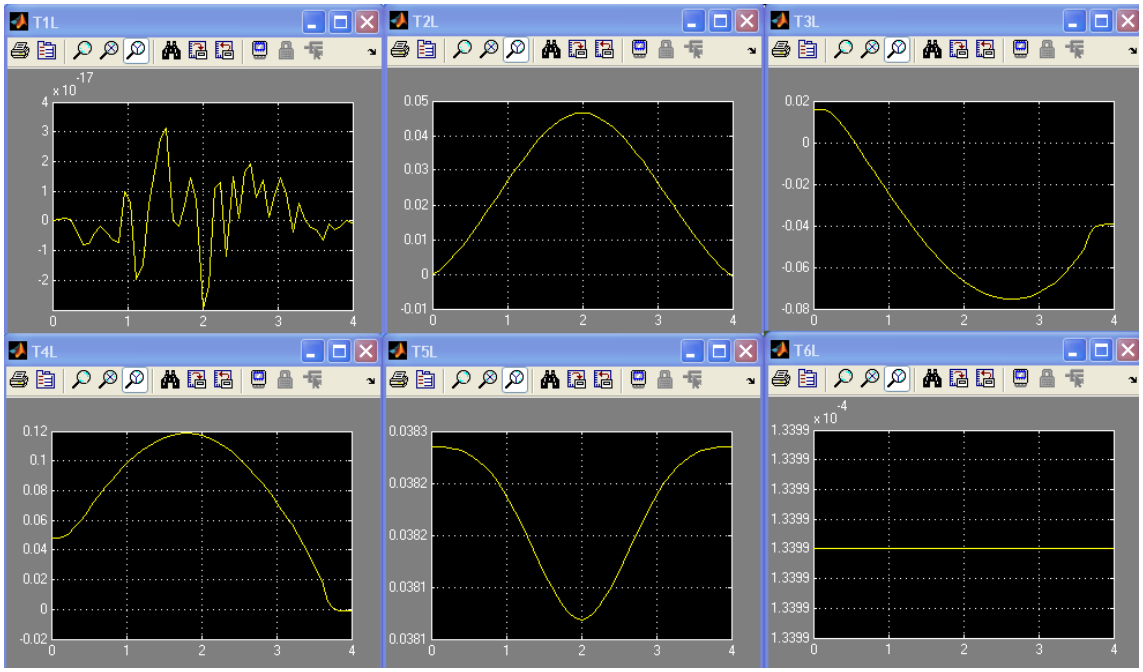


(b) Pares obtenidos con el modelo de SimMechanics

Figura 6.15: Pares de la pierna de soporte



(a) Pares obtenidos con el modelo de Newton-Euler



(b) Pares obtenidos con el modelo de SimMechanics

Figura 6.16: Pares de la pierna flotante

6.3. Caminado

Para las pruebas de caminado se implementaron las trayectorias obtenidas mediante ambos métodos, tanto en el prototipo físico como en el modelo virtual. Estas trayectorias fueron calculadas utilizando un conjunto fijo de parámetros para cada una. Posteriormente se evaluó su estabilidad al obtener la trayectoria del zmp y compararla con el polígono de soporte.

6.3.1. Caminado parametrizado

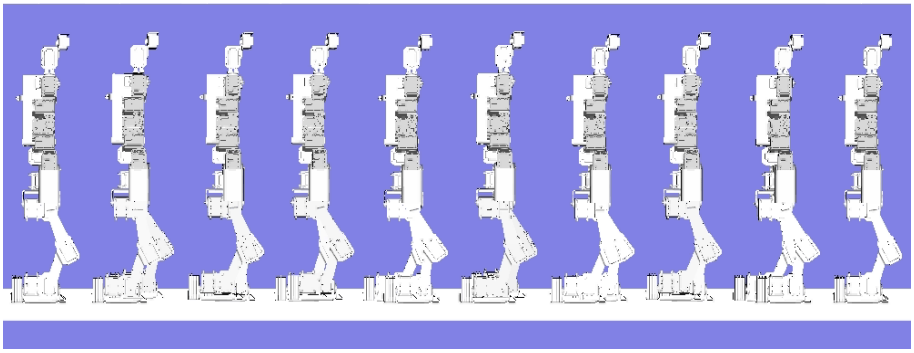
Conjunto de parámetros

- Duración. Tp = 4, Tds = 0.2, deltaT = 0.2
- Plano sagital. Lp = 5, h = 28, hb = 0.5, hp = 1
- Plano frontal. db = -2, dp = 10

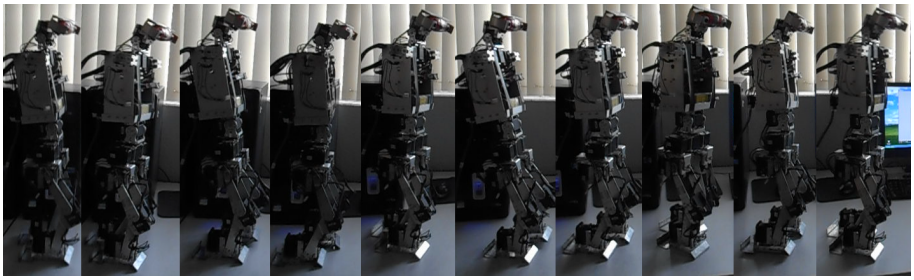
6.3.2. Caminado omnidireccional

Conjunto de parámetros

- Parámetros de entrada. VRobot = [0;1;0], psi = 0.1, l = 29cm
- Acortamiento. vShort = 3, oShort = -0.05
- Carga. vLoad = 2.0
- Balanceo. vSwing = 2.0, oSwing = -0.15, longitud mínima = 0.875

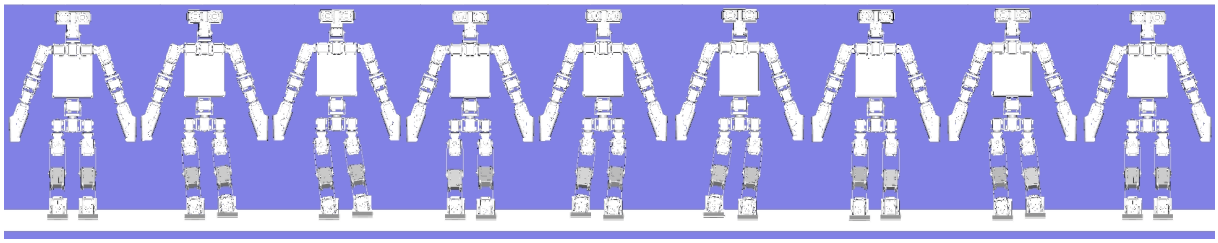


(a) Trayectoria en el modelo VRML

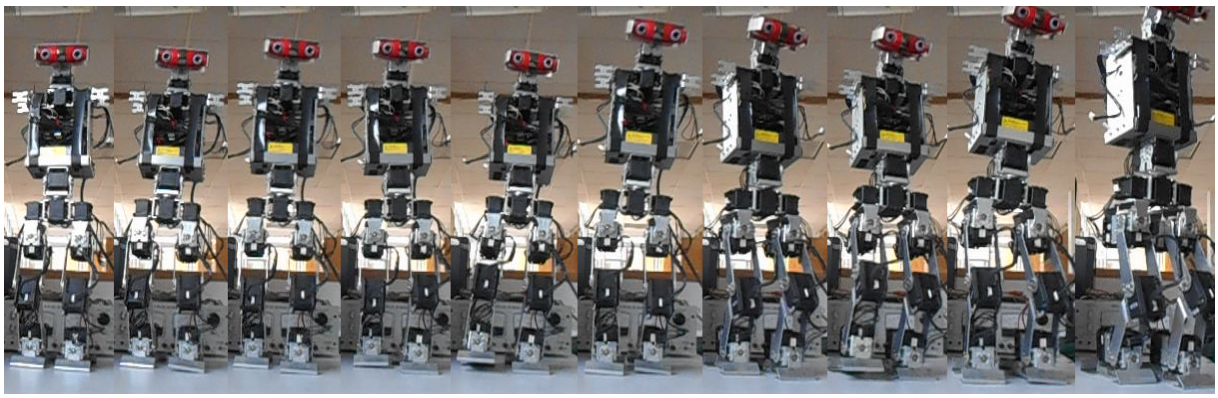


(b) Trayectoria en el prototipo físico

Figura 6.17: Caminado parametrizado en el plano sagital

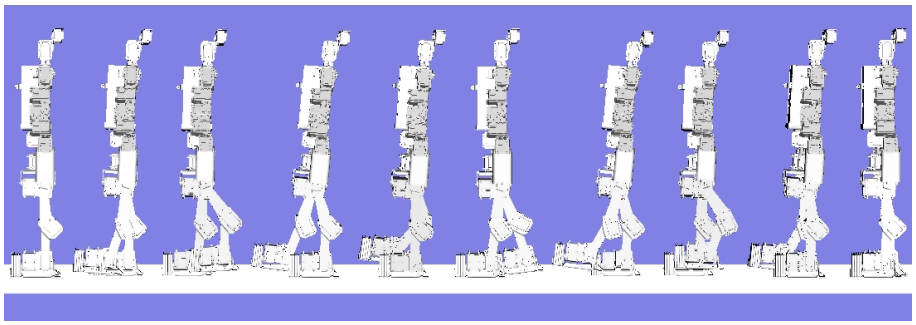


(a) Trayectoria en el modelo VRML

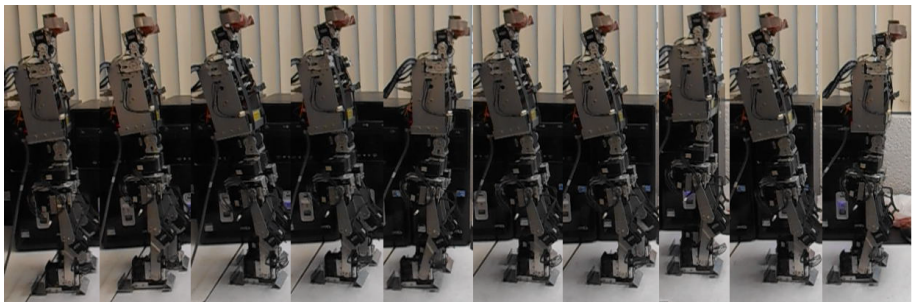


(b) Trayectoria en el prototipo físico

Figura 6.18: Caminado parametrizado en el plano frontal

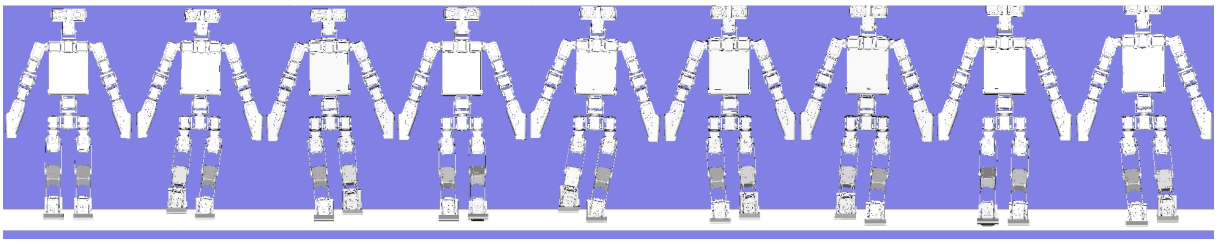


(a) Trayectoria en el modelo VRML

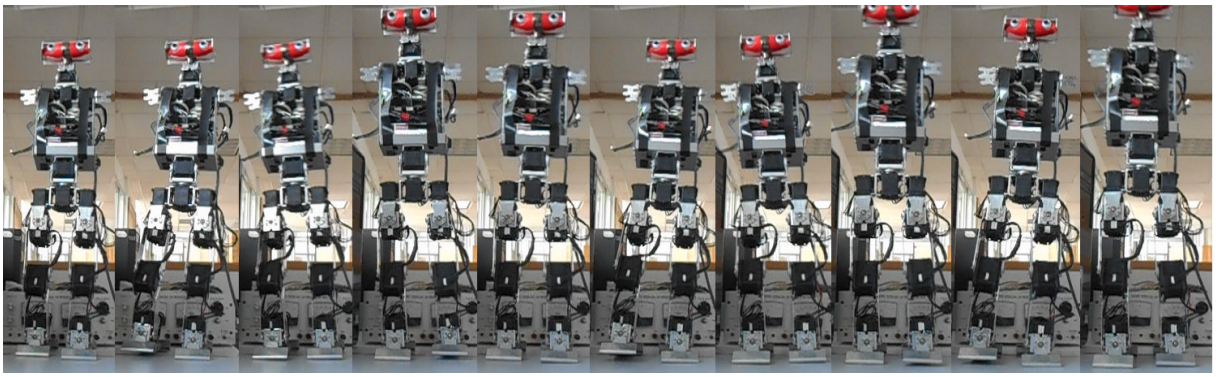


(b) Trayectoria en el prototipo físico

Figura 6.19: Caminado omnidireccional en el plano sagital



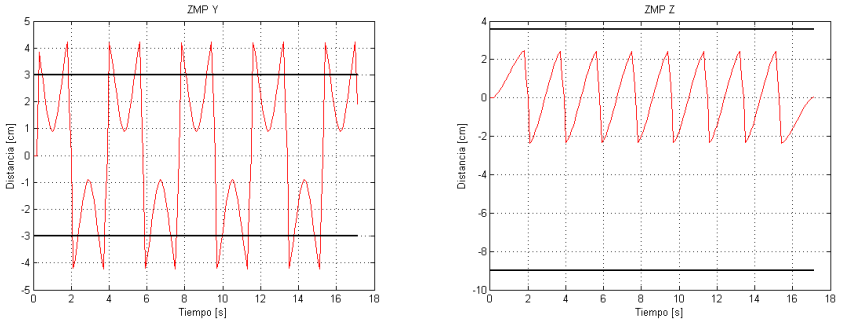
(a) Trayectoria en el modelo VRML



(b) Trayectoria en el prototipo físico

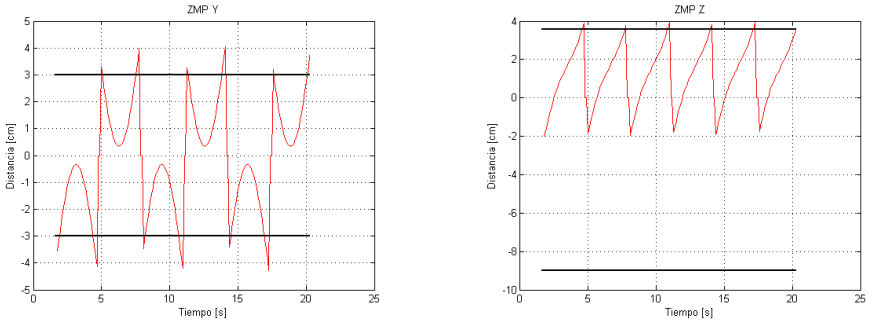
Figura 6.20: Caminado omnidireccional en el plano frontal

6.3.3. Estabilidad del caminado



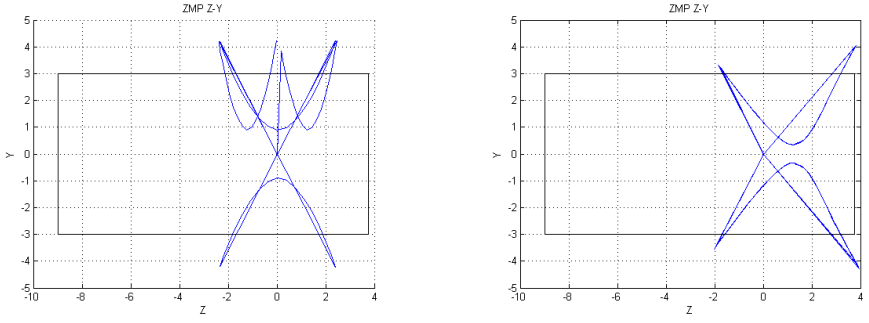
(a) Coordenada Y en el tiempo (b) Coordenada Z en el tiempo

Figura 6.21: Trayectoria del ZMP en el caminado parametrizado



(a) Coordenada Y en el tiempo (b) Coordenada Z en el tiempo

Figura 6.22: Trayectoria del ZMP en el caminado omnidireccional



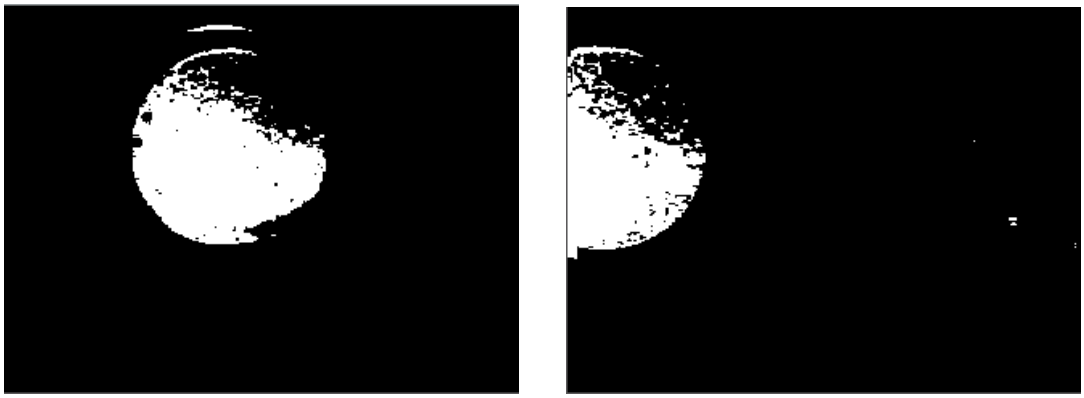
(a) Caminado parametrizado (b) Caminado omnidireccional

Figura 6.23: Trayectoria del ZMP respecto al polígono de soporte simple

6.4. Coordinación Percepción-Control

Para la prueba de coordinación se utilizó la misma pelota roja de la prueba de distancia. Esta pelota se movió dentro del campo de visión de la cámara estereoscópica y mediante los ángulos de *Pan* y de *Tilt* centraba la pelota cuando alguno de estos ángulos fuese significativo. Posteriormente se acercó la pelota y en el momento en que ésta se encontraba al alcance del brazo, el robot la toma y regresa a su posición inicial.

6.4.1. Segmentación



(a) Cámara izquierda (b) Cámara derecha

Figura 6.24: Segmentación de pelota

(a) Cámara izquierda (b) Cámara derecha

Figura 6.25: Detección de pelota

6.4.2. Seguimiento

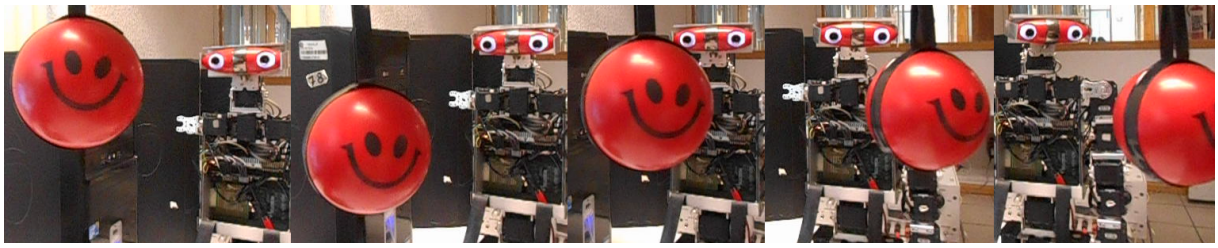


Figura 6.26: Seguimiento de pelota roja utilizando servomotores de Pan y *Tilt*

6.4.3. Alcance de objeto

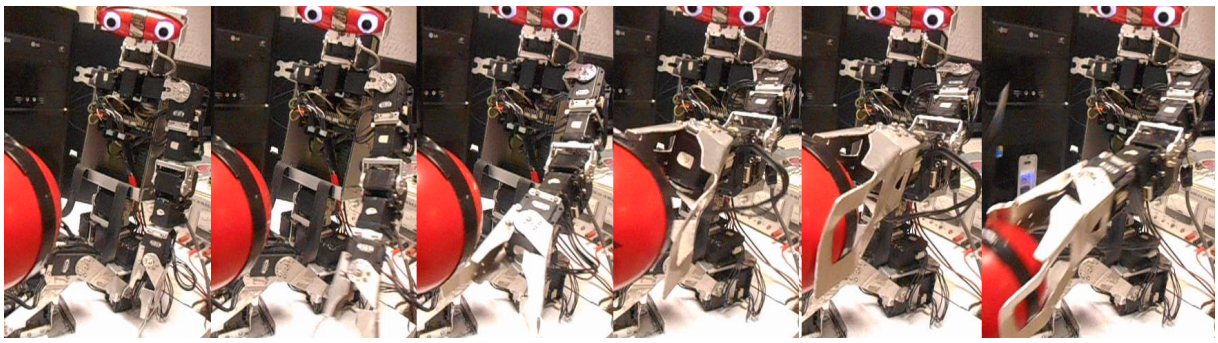


Figura 6.27: Alcance de pelota con brazo izquierdo

Capítulo 7

Conclusiones

En esta Tesis se presentó el análisis que se requiere para llevar a un humanoide desde un estado de reposo hasta un estado de locomoción con cierta autonomía. El proceso descrito es básico para trabajar con cualquier humanoide y es un proceso que muchas veces se da por entendido y se parte del mismo para desarrollar nuevos avances en la robótica de humanoides. Un objetivo de esta Tesis fue describir este proceso básico de análisis para reducir el tiempo que se emplea en conocer los primeros pasos para trabajar con robots humanoides, y así invertirlo en el proceso de investigación e innovación. Este proceso inicial se ilustró mediante la creación de interfaces gráficas que permitieron, tanto validar los algoritmos de cinemática como observar las trayectorias diseñadas para el caminado. La cinemática desarrollada para las piernas se basó en la metodología de Denavit Hartenberg [14] y Lee [16]. La implementación no fue directa pero se basó en el proceso utilizado en su artículo. Por otra parte, la cinemática de los brazos fue desarrollada en su totalidad en esta Tesis. Se descartó la posibilidad de utilizar el algoritmo de Lee debido a que, a pesar de que la solución era exacta, requería de la orientación del efector final y debido al número de gdl del brazo, la orientación depende de la posición, lo cual descarta la posibilidad de este algoritmo. Se utilizó un método geométrico con una compensación numérica que presenta buenos resultados utilizando únicamente la posición de la mano para posicionar el brazo. El modelo dinámico del humanoide fue posible validarlo, de acuerdo a las gráficas mostradas en la Sección 6.2, al comparar los pares con los obtenidos mediante el modelo en SimMechanics, sin embargo, debido a que las trayectorias de SimMechanics no consideran las derivadas de la posición, únicamente la formulación de Newton y la de Euler pudieron

ser comprobadas, el modelo completo sigue sin poder validarse de la misma manera en que se valida el modelo cinemático.

Otro objetivo de esta Tesis fue describir el procedimiento requerido para obtener un humanoide programable que pudiese utilizarse en un futuro para investigación en conjunto con plataformas conocidas como lo son los robots Nao y DARwIn. Este proceso, al igual que el proceso de modelado, se da por conocido debido a que no es necesario cuando se adquiere un humanoide comercialmente. Para este humanoide desarrollado en el Departamento de Control Automático del CINVESTAV en conjunto con la Universidad La Salle, este proceso es fundamental para desarrollar una plataforma básica de investigación. Este proceso puede parecer trivial pero consume demasiado tiempo cuando no se cuenta con un proceso detallado de configuración. Es este proceso el cual se deseó plasmar en este trabajo. Cabe mencionar que el sistema de comunicación implementado con los controladores particulares utilizados en esta Tesis, no exisitía previamente, particularmente el uso del controlador CM-700 con una computadora sin utilizar la terminal de RoboPlus. Sin embargo, este sistema no esta exento del uso del controlador con otra computadora y la terminal de RoboPlus ya que debido al sistema operativo instalado en la Roboard, se requiere utilizar otra computadora para cargar programas al controlador CM-700, además las características del procesador, no permiten instalar el programa que se utiliza para crear programas nuevos para el controlador. Este aspecto del prototipo no lo hace dependiente de otra computadora ya que el programa que se utilizó dentro del controlador sólo sirve de interprete y por lo tanto no requiere modificaciones frecuentes. Lo que si hace dependiente al sistema, de otra computadora es el hecho de que la Roboard no cuenta con salida de video propia a menos de que se utilice la tarjeta de salida VGA. El inconveniente de esto, además de que la tarjeta se calienta rápidamente, es que ocupa el mismo puerto que la tarjeta de red inalámbrica, por lo tanto la computadora se manipula por medio de escritorio remoto. Esto no difiere en gran medida a la forma en que se trabaja con el robot DARwIn ya que requiere de un monitor HD, un teclado y un mouse. El proceso de diseño es un proceso de análisis iterativo en ingeniería. El robot diseñado en esta Tesis fue la segunda iteración del primer humanoide desarrollado en el Departamento de Control Automático. La principal

ventaja sobre el diseño anterior fue la inclusión de baterías y controladores dentro del tórax, lo cual las protege y además facilita las conexiones. El costo de esta mejora fue el aumento del volumen del tórax, el cual se compensó al cambiar el material para no aumentar significativamente el peso del robot. Las piernas no se modificaron ya que sus eslabones eran adecuados para el movimiento del robot y para los servomotores utilizados.

Con respecto a las rutinas de caminado, al implementar las trayectorias en el modelo físico, los servomotores presentaron un error de posición debido a la carga y al acoplamiento mecánico que existe entre el servomotor y la brida de soporte. Esto generó un intervalo en el cual las bridas, podían ser rotadas sin presentar resistencia. Ésta flexibilidad angular fue lo suficientemente considerable para que el robot se balancease hacia adelante o hacia atrás con la misma facilidad, lo cual resultó ser el mayor obstáculo que impedía la implementación de caminatas estables, además de que introdujo un giro en *Yaw* en el pie de soporte. Sin embargo, esta flexibilidad se redujo al colocar acoplamientos en las bridas actuadas de los servomotores, principalmente los del del tobillo. También, se compensó mediante modificaciones en las trayectorias articulares. Esta modificación se puede simplificar mediante la inclusión de sensores que permitan evaluar el seguimiento de las trayectorias de caminado y su estabilidad. Actualmente, se cuenta únicamente con la medición de la posición y la velocidad en los servomotores y se requiere medir más variables para garantizar la estabilidad de cualquier trayectoria implementada. A pesar de haberse mejorado el acoplamiento mecánico, finalmente fue necesario reducir el peso de la unidad pasajera para obtener un caminado estable, ésto se hizo mediante la remoción de los brazos. Ésto se puede evitar en un futuro al colocar un modelo diferente de brazo que posea menos grados de libertad pero que a la vez sea más ligero.

Como último objetivo, se implementó un algoritmo de manipulación de objetos en lazo cerrado mediante un sistema de percepción. Este sistema consiste en: segmentación, estimación de distancia, seguimiento, transformación de coordenadas y posicionamiento del brazo. El paso más importante, y a la vez más difícil de alcanzar, es la segmentación. Todo el algoritmo depende de una correcta detección del objeto y una estimación de la

posición de su centroide. Sin importar la segmentación que utilicemos y los parámetros de la transformada de Hough, esta segmentación no es lo suficientemente robusta para garantizar una correcta detección en todo momento, sin embargo, se establecieron condiciones que se debían cumplir dentro de la detección del objeto antes de posicionar el brazo. Este trabajo abarca los aspectos que se deben tomar como base para el trabajo de investigación en la robótica humanoide, además de que presenta una nueva iteración al proceso de diseño de robots humanoides en el CINVESTAV.

7.1. Trabajo a futuro

Como trabajo a futuro se consideran los siguientes puntos:

- Incorporación de sensores inerciales y de presión
- Manipulación adicional de objetos utilizando el giro de la muñeca mediante el uso de visión y cinemática directa
- Alcance adicional de objetos utilizando el ángulo de *Yaw* del tórax
- Implementación de rutinas de levantado
- Ejecución paralela de algoritmos de visión y caminado
- Modificación de los eslabones en la unidad locomotora para reducir error de posición
- Mejorar el acoplamiento mecánico en la brida de soporte
- Colocar cubiertas al tórax
- Colocar base en los pies que aumente la fricción con el suelo
- Implementación de trayectorias en lazo cerrado midiendo el ZMP para compensar incertidumbre en el modelado y rechazar perturbaciones
- Control de posición basado en la fuerza que se genera al cerrar la mano

- Agregar dispositivos de entrada y salida a la Roboard
- Programas ejecutables de demostración en ausencia de escritorio remoto
- Establecer comunicación bidireccional en los controladores

Apéndice A

Especificaciones del Hardware

Model	CM-700
Processor	ATMega2561
Memory	Flash (Kbytes): 256 Kbytes
ADCs	Analog Devices AD-7918 10-bit
I /O Interface	• Internal I/O Device • Button : 2 sets (Reset 1, Start 1) • Voltage Sensor : 1 • External I/O Device • OLLO Peripheral-Device Compatible 5 pin I/O Port : 6 • 3-Pin Connector for Dynamixel using TTL Communication : 4 • 4-Pin Connector for Dynamixel using RS-485 Communication : 5
Power Consumption	• When IDLE : 40mA • Exterior I/O Maximum Current : 0.9ª • Overall Maximum Current : 10A (Fuse)
Power Input	DC-in 7V to 35V
Dimension	66 x 46 mm
Weight	37.3g

Figura A.1: Especificaciones del subcontrolador CM-700

Model	RB-100
Processor	DM&P Vortex86DX
BIOS	AMI BIOS
Memory	256MB DDR 2 onboard
ADCs	Analog Devices AD-7918 10-bit
I /O Interface	• Micro SD slot x1 **support class 2,4,6 SDHC with any capacity • USB port x 1 (USB 2.0 version)
Connectors	• 2.54 mm 3-pin box header for PWM x 24 • 2.54 mm 10-pin box header for RS-232 x1 • 2.54 mm 10 -pin box header for USB x1 • 2.0 mm 4-pin header for RS-485 x1 • 2.0 mm 4-pin header for TTL serial x1 • 2.54 mm 10 -pin box header for SPI & I 2C x1 • 2.54 mm 16-pin header for A/D x1 • 1.25 mm 3 -pin wafer for TTL serial x 1 • 1.25 mm 4 -pin wafer for LAN x 1 • 1. 2 5 mm 4-pin wafer for MIC-in x 1 • 1. 2 5 mm 4-pin wafer for Line-out x1 • 1. 2 5mm 6-pin wafer for JTAG x1 • 0.8mm 124-Pin Mini PCI Card connector • 3.96 mm 2 pin for Power x 1
Power Consumption	+5V @ 400mA
Power Input	DC-in 6V to 24V
Dimension	96 x 56 mm
Weight	40g
Resolution	PWM : 20ns Serial : 115200bps I2C : 1Kbps ~ 3.3Mbps SPI : 10Mbps to 150Mbps Half-Duplex; CPOL=0/1,CPHA=1 Clock mode
Compatible O/S	DOS, Windows 98/ME, WIndows XP Windows Embedded CE 6.0 Windows XP Embedded, Windows Embedded Stardand Linux distribution kernel 2.4.x, 2.6.x

Figura A.2: Especificaciones de la Roboard RB-100

Model	RX_28
Resolution	0.29°
Gear Reduction Ratio	193 : 1
Stall Torque	3.7N.m (at 18.5V, 1.9A)
No load speed	85rpm (at 18.5V)
Running Degree	• 0° ~ 300° • Endless Turn
Running Temperature	-5℃ ~ +80℃
Command Signal	Digital Packet
Protocol Type	RS485 Asynchronous Serial Communication (8bit,1stop, No Parity)
Link (Physical)	RS485 Multi Drop Bus
ID	254 ID (0~253)
Communication Speed	7343bps ~ 1 Mbps
Feedback	Position, Temperature, Load, Input Voltage, etc.
Power Consumption	Standby current : 50 mA
Power Input	DC-in 12V~18.5V (Recommended Voltage 14.8V)
Material	Full Metal Gear, Engineering Plastic Body
Dimension	35.6 x 50.6 x 35.5 mm
Weight	72g

Figura A.3: Especificaciones del servomotor RX-28

Model	RX_64
Resolution	0.29°
Gear Reduction Ratio	200 : 1
Stall Torque	5.3N.m (at 18.5V, 2.6A)
No load speed	64rpm (at 18.5V)
Running Degree	• 0° ~ 300° • Endless Turn
Running Temperature	-5℃ ~ +80℃
Command Signal	Digital Packet
Protocol Type	RS485 Asynchronous Serial Communication (8bit,1stop, No Parity)
Link (Physical)	RS485 Multi Drop Bus
ID	254 ID (0~253)
Communication Speed	7343bps ~ 1 Mbps
Feedback	Position, Temperature, Load, Input Voltage, etc.
Power Consumption	Standby current : 50 mA
Power Input	DC-in 12V~18.5V (Recommended Voltage 14.8V)
Material	Full Metal Gear, Engineering Plastic Body
Dimension	40.2 x 61.1 x 41 mm
Weight	125g

Figura A.4: Especificaciones del servomotor RX-64

Apéndice B

Configuración de la Roboard

Para trabajar con la Roboard, es necesario instalar un sistema operativo a una tarjeta MicroSD. Se eligió la instalación de Lubuntu debido a que es el sistema que recomienda la página de Roboard y debido a que es un sistema de distribución libre como Ubuntu pero es más ligero, siendo más adecuado para sistemas embebidos. Existen 2 opciones para instalar Lubuntu en la Roboard:

- Utilizando el instalador de Lubuntu directamente en la Roboard.
- Uitlizando la imagen disponible en la imagen de Roboard.

B.1. Instalador de Lubuntu

La página de Roboard tiene diversos tutoriales para la instalación de diferentes sistemas operativos desde el instalador y además cada tutorial tiene diferentes modos de instalación. Estos tutoriales además de estar desactualizados, están incompletos y no condujeron a una instalación exitosa. A continuación se describirá el proceso exitoso de instalación:

1. Descargar una imagen de Lubuntu, en este caso se utilizó la versión 10.04.
2. Crear una USB para instalar Ubuntu, utilizando la imagen de Lubuntu.
3. Colocar una MicroSD con los requerimientos mínimos recomendados. En este caso se utilizó una MicroSD clase 10 de 8 GB de marca SanDisk.

4. Conectar la tarjeta de video a la Roboard y a su vez a un monitor. Además se deben conectar la USB con el instalador, un teclado y un mouse, a los 3 puertos USB disponibles.

5. Encender la Roboard.

6. Seguir las instrucciones de instalación de Ubuntu del tutorial de la página. En este caso no aparecerá el menú de configuración en la pantalla inicial y esta parte del tutorial no se podrá realizar pero esto no afectará la instalación de Lubuntu. Las particiones de la memoria se realizan de la misma manera que en el tutorial. El tutorial se sigue hasta la compilación de la imagen de Linux que se encuentra disponible en la página de la Roboard.

7. Una vez que se termina el tutorial de instalación de Ubuntu y se reinicia la Roboard, se debe configurar la conexión de Internet abriendo las conexiones y creando una nueva conexión vía Ethernet por defecto. Esta conexión se configura con los datos particulares del dominio y la ip de la red a la que se conecta y además se debe de cambiar el puerto Ethernet por defecto al puerto 8.

8. Ahora se debe seguir el tutorial de la imagen de Lubuntu para habilitar los puertos que tiene la Roboard y para habilitar el uso de escritorio remoto ya que la tarjeta de video se calienta. Esta parte de los tutoriales está correcta.

Este proceso de instalación tiene un inconveniente al compilar el kernel con los archivos provistos por la página de Roboard. Esto es que al momento de iniciar nuevamente la Roboard, aparece una pantalla de Ubuntu además de la pantalla de Lubuntu pero esto no afecta el funcionamiento del sistema.

B.2. Imagen de Lubuntu

1. Descargar la imagen de Lubuntu proporcionada por la página de Roboard. Esta imagen actualmente no está disponible desde el hipervínculo de la página, la

imagen se encuentra en un vínculo que aparece en un foro de RoboSavvy `http://robosavvy.com/forum/viewtopic.php?t=7347`.

2. Seguir el tutorial para instalar la imagen en la Roboard. Este tutorial es el único completo y correcto. El único problema con el tutorial es que al iniciar la Roboard y seleccionar la opción de instalación del menú de Clonezilla, la Roboard muestra una pantalla negra y si después de unos segundos no aparece el proceso de instalación, debe reiniciarse la Roboard hasta que aparezca.

3. El proceso de configuración de la LAN puede realizarse una vez iniciada la sesión en ambiente gráfico sin necesidad de utilizar la terminal, tal como se describe en el procedimiento anterior.

4. Por último debe habilitarse el uso de escritorio remoto.

Un punto importante de la configuración de Lubuntu es que no se debe tratar de modificar la resolución de la pantalla cuando se utiliza la tarjeta de video. Esto es porque a pesar de que existen numerosas soluciones, ninguna logra un cambio exitoso y además pueden alterar las opciones de la interfaz gráfica evitando el acceso a Lubuntu la siguiente vez que se encienda la Roboard. Además, al habilitar el escritorio remoto, se deja de trabajar con la resolución de la tarjeta de video de la Roboard.

Apéndice C

Instalación de OpenCV

El siguiente procedimiento está basado en el proceso descrito en el tutorial de OzBotz `http://www.ozbotz.org/opencv-installation/` con algunas modificaciones.

1. Se deben realizar los primeros 10 pasos del tutorial. Cuando no se encuentran todas las aplicaciones mencionadas en los pasos, se deben buscar opciones similares como por ejemplo *git* y *libjpeg8*. Para esto se recomienda instalar Synapse para buscar e instalar aplicaciones. En caso de no encontrarlas omitirlas.

2. Antes de bajar OpenCV, se deben instalar los siguientes paquetes manualmente:

   - libjpeg62-dev
   - libtiff4-dev
   - zlib1g-dev
   - libjasper-dev
   - libavcodec-dev
   - libdc1394-22-dev
   - libgstreamer0.10-dev
   - libgstreamer-plugins-base0.10-dev
   - libavformat-dev
   - libv4l-dev
   - libswscale-dev

3. Descargar la versión de OpenCV 2.4.3 y continuar con el paso 10 hasta *cd build*. Para compilar la carpeta en vez de utilizar la línea del tutorial, esta se reemplazará por *cmake -D CMAKE_BULD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_PYTHON_SUPPORT=ON USE_V4L=ON WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON USE_ GStreamer=ON ..*

4. Una vez que finaliza, se verifica que dentro del resumen se encuentren las opciones tal como se muestran en el tutorial, de lo contrario se deben revisar los paquetes que muestran error ya que al ejecutar el comando *make*, tardará alrededor de 5 horas y 30 minutos para compilar la carpeta y si se presenta un error, se tendrá que corregir y se tendrá que iniciar de nuevo el proceso de compilación.

5. Una vez que termina la ejecución del comando *make* seguir el tutorial hasta el final. Un punto importante que se debe tomar en cuenta antes de utilizar cualquier cámara, es que se debe verificar los permisos para utilizarla, en caso de obtener cuadros vacíos se debe agregar el usuario a un grupo de video y además verificar que la cámara está siendo detectada por Linux dentro de los dispositivos.

Apéndice D

Conversión de Datos: double-unsigned char-double

1. Se utiliza la función *sprintf* para convertir un dato *double* a string.
2. Se divide el arreglo de caracteres en dos arreglos, uno para la parte entera y otro para los primeros dos decimales.
3. Se convierte cada arreglo a entero utilizando la función *atoi*.
4. Se utilizan estos enteros como caracteres sin signo. En este punto se convierten los valores negativos a un valor entre 1 y 255.
5. Los caracteres sin signo se convierten a enteros y si un número entero es mayor de 130, se resta 255 ya que se encuentra dentro de la parte que involucra números negativos.
6. Los números enteros se convierten a *string* utilizando la función *sprintf*.
7. Se concatena la parte entera y decimal en un mismo arreglo utilizando la función *strcat*.
8. Por último se convierte el arreglo a tipo *double* con la función *atof*.

Apéndice E

Código VRML

E.1. Unidad Pasajera

```
DEF UPasajera Group {
  children [

    Jabdominal {}

    Group {
      children [
        DEF JABD_Sensor CylinderSensor {
          diskAngle 0.7854
          minAngle -3.0543
          maxAngle  3.0543
        }

        DEF JABD Transform {
          children [

            DEF AABD Transform  {
              rotation 0 1 0 -1.5708
              translation    0.1674  0 0        #0.1135
              children [

                Cuerpo {}
                #RefB1 {}

                DEF ABALL Transform {
                  rotation 0.7071 0.7071 0 3.1416
                  children [

                    DEF JBALL Transform {
                      children [

                        Pelota {}

                      ]
                    }
                  ]
                }

                DEF Cabeza Group {
                  children [

                    DEF AUXH Transform {
                      rotation 1 0 0 -1.5708
                      translation    0 0 0
                      children [

                        DEF JPAN_Sensor CylinderSensor {
                          diskAngle 0.7854
                          minAngle -3.0543
                          maxAngle  3.0543
                        }
                        DEF JPAN Transform {
                          children [

                            DEF APAN Transform {
                              rotation 0.5773 0.5773 0.5773 2.0943 #R(y,90)R(z,90)
                              translation    0 0.04675 0
                              children [

                                BR28h {}

```

```
                        DEF JTILT_Sensor CylinderSensor {
                          diskAngle 0.7854
                          minAngle -3.0543
                          maxAngle  3.0543
                        }

                        DEF JTILT Transform {
                          children [

                            DEF ATILT Transform {
                              rotation 0.5773 -0.5773 0.5773 2.0943 #R(y,-90)R(z,90)
                              translation    0 0 0
                              children [

                                Cabeza {}

                                DEF REYE Viewpoint {
                                  description    "RIGHT_EYE"
                                  position     0.036416 -0.048 -0.028
                                  orientation    0.7071 0 -0.7071 3.1416  #R(x,180)R(y,90)
                                  fieldOfView    0.78 #2.2
                                }

                                DEF LEYE Viewpoint {
                                  description    "LEFT_EYE"
                                  position     0.036416 -0.048 0.028
                                  orientation    0.7071 0 -0.7071 3.1416  #R(x,180)R(y,90)
                                  fieldOfView    0.78    #2.2
                                }
                              ]
                            }
                          ]
                        }
                      ]
                    }
                  ]
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}

DEF RArm Group {
  children [

    DEF J1RA_Sensor CylinderSensor {
      diskAngle 0.7854
      minAngle -3.0543
      maxAngle  3.0543
    }

    DEF J1RA Transform {
      children [

        DEF A1RA Transform {
          rotation  0 0 1 1.5708
          translation    0 0.08715 0
          children [

            BR28h {}

            DEF J2RA_Sensor CylinderSensor {
              diskAngle 0.7854
              minAngle -3.0543
              maxAngle  3.0543
            }

            DEF J2RA Transform {
              children [

                DEF A2RA Transform {
                  rotation  0 0.7071 -0.7071 3.1416  #R(z,180)R(x,-90)
                  translation    0 0 0.083
                  children [

                    Hombro {}

                    DEF J3RA_Sensor CylinderSensor {
                      diskAngle 0.7854
                      minAngle -3.0543
                      maxAngle  3.0543
                    }

                    DEF J3RA Transform  {
                      children [

                        DEF A3RA Transform {
                          rotation   0.7071 -0.7071 0 3.1416  #R(x,180)R(z,90)
                          translation    0 -0.0255 0
                          children [

                            BR28h {}
```

```
                                    DEF J4RA_Sensor CylinderSensor {
                                      diskAngle 0.7854
                                      minAngle -3.0543
                                      maxAngle  3.0543
                                    }

                                    DEF J4RA Transform {
                                      children [

                                        DEF A4RA Transform {
                                          rotation   0.5773 -0.5773 0.5773 2.0943  #R(y,90)R(z,90)
                                          translation    0.0609 0 0
                                          children [

                                            Antebrazo {}

                                            DEF J5RA_Sensor CylinderSensor {
                                              diskAngle 0.7854
                                              minAngle -3.0543
                                              maxAngle  3.0543
                                            }

                                            DEF J5RA Transform {
                                              children [

                                                DEF A5RA Transform {
                                                  rotation 0 1 0 1.5708
                                                  translation    0 -0.0415 0
                                                  children [

                                                    ManoD {}

                                                    DEF ARBALL Transform    {
                                                      #translation  0 -0.0205 0.07
                                                      translation    0 -0.05 0  #axis to gripper center

                                                      children [

                                                        PelotaM {}

                                                      ]
                                                    }
                                                  ]
                                                }
                                              ]
                                            }
                                          ]
                                        }
                                      ]
                                    }
                                  ]
                                }
                              ]
                            }
                          ]
                        }
                      ]
                    }
                  ]
                }
              ]
            }
          ]
        }
      ]
    }

    DEF LArm Group {
      children [

        DEF J1LA_Sensor CylinderSensor {
          diskAngle 1.5708
          minAngle -3.0543
          maxAngle  3.0543
        }

        DEF J1LA Transform {
          children [

            DEF A1LA Transform {
              rotation 0.7071 -0.7071 0 3.1416
              translation    0 -0.08715 0
              children [

                BR28h {}

                DEF J2LA_Sensor CylinderSensor {
                  diskAngle 0.7854
                  minAngle -3.0543
                  maxAngle  3.0543
                }

                DEF J2LA Transform {
                  children [
```

```
247
248                                 DEF A2LA Transform {
249                                   rotation 1 0 0 1.5708
250                                   translation    0 0 -0.08275
251                                   children [
252
253                                     Hombro {}
254
255                                     DEF J3LA_Sensor CylinderSensor {
256                                       diskAngle 0.7854
257                                       minAngle -3.0543
258                                       maxAngle  3.0543
259                                     }
260
261                                     DEF J3LA Transform  {
262                                       children [
263
264                                         DEF A3LA Transform {
265                                           rotation  0.7071 -0.7071 0 3.1416  #R(x,180)R(z,90)
266                                           translation    0 -0.0255 0
267                                           children [
268
269                                             BR28h {}
270
271                                             DEF J4LA_Sensor CylinderSensor {
272                                               diskAngle 0.7854
273                                               minAngle -3.0543
274                                               maxAngle  3.0543
275                                             }
276
277                                             DEF J4LA Transform {
278                                               children [
279                                                 DEF A4LA Transform {
280                                                   rotation  0.5773 -0.5773 0.5773 2.0943  #R(y,90)R(z,90)
281                                                   translation    0.0609 0 0
282                                                   children [
283
284                                                     Antebrazo {}
285
286                                                     DEF J5LA_Sensor CylinderSensor {
287                                                       diskAngle 0.7854
288                                                       minAngle -3.0543
289                                                       maxAngle  3.0543
290                                                     }
291
292                                                     DEF J5LA Transform {
293                                                       children [
294
295                                                         DEF A5LA Transform {
296                                                           rotation 0 1 0 1.5708
297                                                           translation    0 -0.0415 0
298                                                           children [
299
300                                                             ManoI {}
301
302                                                             DEF ALBALL Transform    {
303
304                                                               #translation  0 -0.0205 0.07
305                                                               translation    0 -0.05 0  #axis to grip center
306                                                               children [
307
308                                                                 PelotaM {}
309
310                                                               ]
311                                                             }
312                                                           ]
313                                                         }
314                                                       ]
315                                                     }
316                                                   ]
317                                                 }
318                                               ]
319                                             }
320                                           ]
321                                         }
322                                       ]
323                                     }
324                                   ]
325                                 }
326                               ]
327                             }
328                           ]
329                         }
330                       ]
331                     }
332                   ]
333                 }
334               ]
335             }
336           ]
337         }
338       ]
```

```
339       }
340     ]
341   }
```

E.2. Unidad Locomotora

```
482   DEF Mode Switch {
483
484       whichChoice 0
485       choice [
486
487           DEF BaseB Group {   #Modo 0
488             children [
489               JCad {}
490               #RefB2 {}
491
492               Group {
493                 children [
494                   DEF UBody Group {
495                     children [
496                       DEF AUP Transform {
497                         translation    0 0.027 0
498                         children [
499                           UPasajera {}
500                         ]
501                       }
502                     ]
503                   }
504
505                   DEF RLeg Group {
506                     children [
507                       DEF ACRL Transform {
508                         rotation 0 1 0 3.1415
509                         translation    0 -0.02125 0.0452
510                         #translation  0 0 0.0452
511                         children [
512                           #JCadR  {}
513
514                           Group {
515                             children [
516                               DEF J1RL_Sensor CylinderSensor {
517                                 diskAngle 0.7854
518                                 minAngle -3.0543
519                                 maxAngle  3.0543
520                               }
521
522                               DEF J1RL Transform {
523                                 children [
524                                   DEF A1JRL Transform {
525                                     #rotation 1 0 0 1.5708
526                                     rotation 0.57735 -0.57735 0.57735 2.094395
527                                     translation    0 -0.031 0
528                                     children [
529
530                                       Cadera {}
531
532                                       Group {
533                                         children [
534
535                                           DEF J2RL_Sensor CylinderSensor {
536                                             diskAngle 0.7854
537                                             minAngle -3.0543
538                                             maxAngle  3.0543
539                                           }
540
541                                           DEF J2RL Transform {
542                                             children [
543                                               DEF A2RL Transform {
544                                                 rotation -0.57735 -0.57735 -0.57735 2.094395      # TH = 120 R [010;001;100]
545                                                 translation    0 0 0
546                                                 children [
547
548                                                   JAcoplados {}
549
550                                                   Group {
551                                                     children [
552                                                       DEF J3RL_Sensor CylinderSensor  {
553                                                         diskAngle 0.7854
554                                                         minAngle -3.0543
555                                                         maxAngle  3.0543
556                                                       }
557
558                                                       DEF J3RL Transform {
559                                                         children [
560                                                           DEF A3RL Transform {
561                                                             rotation 0 1 0 3.141592
```

```
                                    #translation  0.094553 0 0
                                    translation    0.095659 0 0
                                    children [

                                      RodillaTobillo {}

                                      Group {
                                        children [
                                          DEF J4RL_Sensor CylinderSensor {
                                            diskAngle 0.7854
                                            minAngle -3.0543
                                            maxAngle  3.0543
                                          }

                                          DEF J4RL  Transform {
                                            children [
                                              DEF A4RL Transform {
                                                rotation 0 1 0 1.5708
                                                translation    -0.114207 0 0
                                                children [

                                                  JRT {}

                                                  Group {
                                                    children [
                                                      DEF J5RL_Sensor CylinderSensor {
                                                        diskAngle 0.7854
                                                        minAngle -3.0543
                                                        maxAngle  3.0543
                                                      }

                                                      DEF J5RL Transform {
                                                        children [
                                                          DEF A5RL Transform {
                                                            rotation 0 0 1 -1.5708
                                                            translation    0 0 0
                                                            children [

                                                              JAcopladosT {}

                                                              Group {

                                                                children [
                                                                  DEF J6RL_Sensor CylinderSensor {
                                                                    diskAngle 0.7854
                                                                    minAngle -3.0543
                                                                    maxAngle  3.0543
                                                                  }

                                                                  DEF J6RL Transform {
                                                                    children [
                                                                      DEF A6RL Transform {
                                                                        rotation 0 0 0 0
                                                                        translation    0 -0.002 -0.02775
                                                                        children [
                                                                          Pie {}
                                                                        ]
                                                                      }
                                                                    ]
                                                                  }
                                                                ]
                                                              }
                                                            ]
                                                          }
                                                        ]
                                                      }
                                                    ]
                                                  }
                                                ]
                                              }
                                            ]
                                          }
                                        ]
                                      }
                                    ]
```

```
                    }
                  ]
                }
              ]
            }

            DEF LLeg Group {
              children [
                DEF ACLL Transform {
                  rotation 0 1 0 3.1415
                  translation    0 -0.02125 -0.0452
                  children [

                    #JCadL {}

                    Group {
                      children [
                        DEF J1LL_Sensor CylinderSensor {
                          diskAngle 0.7854
                          minAngle -3.0543
                          maxAngle  3.0543
                        }

                        DEF J1LL Transform {
                          children [
                            DEF AJ1LL Transform {
                              #rotation 1 0 0 1.5708
                              rotation 0.57735 -0.57735 0.57735 2.094395
                              translation    0 -0.031 0
                              children [

                                Cadera {}

                                Group {
                                  children [
                                    DEF J2LL_Sensor CylinderSensor {
                                      diskAngle 0.7854
                                      minAngle -3.0543
                                      maxAngle  3.0543
                                    }

                                    DEF J2LL Transform {
                                      children [
                                        DEF A2LL Transform {
                                          rotation -0.57735 -0.57735 -0.57735 2.094395      # TH = 120 R [010;001;100]
                                          translation    0 0 0
                                          children [

                                            JAcoplados {}

                                            Group {
                                              children [
                                                DEF J3LL_Sensor CylinderSensor {
                                                  diskAngle 0.7854
                                                  minAngle -3.0543
                                                  maxAngle  3.0543
                                                }

                                                DEF J3LL    Transform {
                                                  children [
                                                    DEF A3LL Transform {
                                                      rotation 0 1 0 3.141592
                                                      #translation  0.094553 0 0
                                                      translation    0.095659 0 0
                                                      children [

                                                        RodillaTobillo {}

                                                        Group {
                                                          children [
                                                            DEF J4LL_Sensor CylinderSensor {
                                                              diskAngle 0.7854
                                                              minAngle -3.0543
                                                              maxAngle  3.0543
                                                            }

                                                            DEF J4LL Transform {
                                                              children [
                                                                DEF A4LL Transform {
                                                                  rotation 0 1 0 1.5708
                                                                  translation    -0.114207 0 0
                                                                  children [

                                                                    JRT {}

                                                                    Group {
                                                                      children [
                                                                        DEF J5LL_Sensor CylinderSensor {
                                                                          diskAngle 0.7854
                                                                          minAngle -3.0543
                                                                          maxAngle  3.0543
                                                                        }
```

```
                                                                    DEF J5LL Transform {
                                                                      children [
                                                                        DEF A5LL Transform {
                                                                          rotation 0 0 1 −1.5708
                                                                          translation    0 0 0
                                                                          children [

                                                                            JAcopladosT {}

                                                                            Group {
                                                                              children [
                                                                                DEF J6LL_Sensor CylinderSensor {
                                                                                  diskAngle 0.7854
                                                                                  minAngle −3.0543
                                                                                  maxAngle  3.0543
                                                                                }

                                                                                DEF J6LL Transform {
                                                                                  children [
                                                                                    DEF A6LL Transform {
                                                                                      rotation 0 0 0 0
                                                                                      translation    0 −0.002 −0.02775
                                                                                      children [

                                                                                        Pie {}
                                                                                      ]
                                                                                    }
                                                                                  ]
                                                                                }
                                                                              ]
                                                                            }
                                                                          ]
                                                                        }
                                                                      ]
                                                                    }
                                                                  ]
                                                                }
                                                              ]
                                                            }
                                                          ]
                                                        }
                                                      ]
                                                    }
                                                  ]
                                                }
                                              ]
                                            }
                                          ]
                                        }
                                      ]
                                    }
                                  ]
                                }
                              ]
                            }
                          ]
                        }
                      ]
                    }
                  ]
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}


DEF BaseRF Group {
  children [
    DEF RF Transform {
      children [
        DEF ARF Transform {
          rotation −0.57735 0.57735 0.57735 2.094395
          #translation  0 −0.263153 0.0452
          translation    0 −0.262116 0.0452  #considerando 95.659 de rodilla_tobillo

          children [

            PieRev {}

            Group {
              children [
                DEF J6RLRev_Sensor CylinderSensor {
                  diskAngle 0.7854
                  minAngle −3.0543
                  maxAngle  3.0543
```

```
836                     }
837
838                     DEF J6RLRev Transform {
839                       children [
840                         DEF A6RLRev Transform {
841                           rotation 0.5773 -0.5773 0.5773 2.0943
842                           translation    0 0 0
843                           children [
844
845                             JAcopladosTRev {}
846
847                             Group {
848                               children [
849                                 DEF J5RLRev_Sensor CylinderSensor {
850                                   diskAngle 0.7854
851                                   minAngle -3.0543
852                                   maxAngle  3.0543
853                                 }
854
855                                 DEF J5RLRev Transform {
856                                   children [
857                                     DEF A5RLRev Transform {
858                                       rotation 0 1 0 -1.5708
859                                       translation    0.114207 0 0
860                                       children [
861
862                                         JRTRev {}
863
864                                         Group {
865                                           children [
866                                             DEF J4RLRev_Sensor CylinderSensor {
867                                               diskAngle 0.7854
868                                               minAngle -3.0543
869                                               maxAngle  3.0543
870                                             }
871
872                                             DEF J4RLRev Transform {
873                                               children [
874                                                 DEF A4RLRev Transform {
875                                                   rotation 0 1 0 1.5708
876                                                   #translation  0.0011264 0 -0.096696
877                                                   #translation  0 0 -0.0955696
878                                                   translation    0 0 -0.095659
879                                                   children [
880
881                                                     RodillaTobilloRev {}
882
883                                                     Group {
884                                                       children [
885                                                         DEF J3RLRev_Sensor CylinderSensor {
886                                                           diskAngle 0.7854
887                                                           minAngle -3.0543
888                                                           maxAngle  3.0543
889                                                         }
890
891                                                         DEF J3RLRev Transform {
892                                                           children [
893                                                             DEF A3RLRev Transform {
894                                                               rotation -0.5773 0.5773 -0.5773 2.0943
895                                                               translation 0 0 0
896                                                               children [
897
898                                                                 JAcopladosRev {}
899
900                                                                 Group {
901                                                                   children [
902                                                                     DEF J2RLRev_Sensor CylinderSensor {
903                                                                       diskAngle 0.7854
904                                                                       minAngle -3.0543
905                                                                       maxAngle  3.0543
906                                                                     }
907
908                                                                     DEF J2RLRev Transform {
909                                                                       children [
910                                                                         DEF A2RLRev Transform {
911                                                                           rotation -0.5773 0.5773 -0.5773 2.0943
912                                                                           translation    0 0 0.031
913                                                                           children [
914
915                                                                             CaderaRev {}
916
917                                                                             Group {
918                                                                               children [
919                                                                                 DEF J1RLRev_Sensor CylinderSensor {
920                                                                                   diskAngle 0.7854
921                                                                                   minAngle -3.0543
922                                                                                   maxAngle  3.0543
923                                                                                 }
924
925                                                                                 DEF J1RLRev Transform {
926                                                                                   children [
927                                                                                     DEF A1RLRev Transform {
928                                                                                       rotation 1 0 0 3.1415
```

```
                                                                                translation    0 -0.02125 0.0452
                                                                                children [

                                                                                  JCad {}
                                                                                  RefB2 {}

                                                                                  USE UBody
                                                                                  USE LLeg
                                                                                ]
                                                                              }
                                                                            ]
                                                                          }
                                                                        ]
                                                                      }
                                                                    ]
                                                                  }
                                                                ]
                                                              }
                                                            ]
                                                          }
                                                        ]
                                                      }
                                                    ]
                                                  }
                                                ]
                                              }
                                            ]
                                          }
                                        ]
                                      }
                                    ]
                                  }
                                ]
                              }
                            ]
                          }
                        ]
                      }
                    ]
                  }
                ]
              }
            ]
          }
        ]
      }

      DEF BaseLF Group {
        children [
          DEF LF Transform {
            children [
              DEF ALF Transform {
                rotation -0.57735 0.57735 0.57735 2.094395
                #translation  0 -0.263153 -0.0452
                translation    0 -0.262116 -0.0452
                children [

                  PieRev {}

                  Group {
                    children [
                      DEF J6LLRev_Sensor CylinderSensor {
                        diskAngle 0.7854
                        minAngle -3.0543
                        maxAngle  3.0543
                      }

                      DEF J6LLRev Transform {
                        children [
                          DEF A6LLRev Transform {
                            rotation 0.5773 -0.5773 0.5773 2.0943
                            translation    0 0 0
                            children [

                              JAcopladosTRev {}

                              Group {
                                children [
                                  DEF J5LLRev_Sensor CylinderSensor {
                                    diskAngle 0.7854
                                    minAngle -3.0543
                                    maxAngle  3.0543
                                  }

                                  DEF J5LLRev Transform {
                                    children [
```

```
                                    DEF A5LLRev Transform {
                                      rotation 0 1 0 -1.5708
                                      translation    0.114207 0 0
                                      children [

                                        JRTRev {}

                                        Group {
                                          children [
                                            DEF J4LLRev_Sensor CylinderSensor {
                                              diskAngle 0.7854
                                              minAngle -3.0543
                                              maxAngle  3.0543
                                            }

                                            DEF J4LLRev Transform {
                                              children [
                                                DEF A4LLRev Transform {
                                                  rotation 0 1 0 1.5708
                                                  #translation  0 0 -0.096696
                                                  translation    0 0 -0.095659
                                                  children [

                                                    RodillaTobilloRev {}

                                                    Group {
                                                      children [
                                                        DEF J3LLRev_Sensor CylinderSensor {
                                                          diskAngle 0.7854
                                                          minAngle -3.0543
                                                          maxAngle  3.0543
                                                        }

                                                        DEF J3LLRev Transform {
                                                          children [
                                                            DEF A3LLRev Transform {
                                                              rotation -0.5773 0.5773 -0.5773 2.0943
                                                              translation    0 0 0
                                                              children [

                                                                JAcopladosRev {}

                                                                Group {
                                                                  children [
                                                                    DEF J2LLRev_Sensor CylinderSensor {
                                                                      diskAngle 0.7854
                                                                      minAngle -3.0543
                                                                      maxAngle  3.0543
                                                                    }

                                                                    DEF J2LLRev Transform {
                                                                      children [
                                                                        DEF A2LLRev Transform {
                                                                          rotation -0.5773 0.5773 -0.5773 2.0943
                                                                          translation    0 0 0.031
                                                                          children [

                                                                            CaderaRev {}

                                                                            Group {
                                                                              children [
                                                                                DEF J1LLRev_Sensor CylinderSensor {
                                                                                  diskAngle 0.7854
                                                                                  minAngle -3.0543
                                                                                  maxAngle  3.0543
                                                                                }

                                                                                DEF J1LLRev Transform {
                                                                                  children [
                                                                                    DEF A1LLRev Transform {
                                                                                      rotation 1 0 0 3.1415
                                                                                      translation    0 -0.02125 -0.0452
                                                                                      children [

                                                                                        JCad {}
                                                                                        RefB2 {}

                                                                                        USE UBody
                                                                                        USE RLeg
                                                                                      ]
                                                                                    }
                                                                                  ]
                                                                                }
                                                                              ]
                                                                            }
                                                                          ]
                                                                        }
                                                                      ]
                                                                    }
                                                                  ]
                                                                }
                                                              ]
                                                            }
                                                          ]
                                                        }
                                                      ]
                                                    }
                                                  ]
                                                }
                                              ]
                                            }
                                          ]
                                        }
                                      ]
                                    }
```

```
                                                    }
                                                ]
                                            }
                                        ]
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        ]
    }
```

Apéndice F

Código Matlab

F.1. Cinemática Piernas

```
clear all
close all
clc

%% Angulos y Eslabones

t1 = 10
t2 = 20
t3 = 30
t4 = 40
t5 = 50
t6 = 60

LL1 = 4.52;
LL2 = 5.225;
LL3 = 9.6696;
LL4 = 11.4207;
LL5 = 2.775;

% Pierna
L = 1;
%% Cinematica Directa

A1 = mpaso(t1,90,0,0);
A2 = mpaso(t2-90,-90,0,0);
A3 = mpaso(t3,0,LL3,0);
A4 = mpaso(t4,0,LL4,0);
A5 = mpaso(t5,90,0,0);
A6 = mpaso(t6,0,LL5,0);

T0_6 = A1*A2*A3*A4*A5*A6;

T6_0 = inv(T0_6);

A0 = [-1  0 0  L*4.52;
       0 -1 0    0;
       0  0 1 -5.225;
       0  0 0    1];      % B2_0

TB2_6 = A0*A1*A2*A3*A4*A5*A6;

%% Cinematica Inversa
pp = T6_0(1:3,4);
ppx = T6_0(1,4);
ppy = T6_0(2,4);
ppz = T6_0(3,4);

aax = T6_0(1,3);
aay = T6_0(2,3);
aaz = T6_0(3,3);

ssx = T6_0(1,2);
ssy = T6_0(2,2);
ssz = T6_0(3,2);

nnx = T6_0(1,1);
nny = T6_0(2,1);
nnz = T6_0(3,1);
%% TH4

C4 = ((ppx+LL5)^2+ppy^2+ppz^2-LL3^2-LL4^2)/(2*LL3*LL4);
```

```
62
63  KNEE = 1;
64
65  if KNEE == 0
66      KNEE = 1;
67  end
68
69  if(C4>1)
70      TH4 = atan2(0,C4);
71  else
72      TH4 = atan2(KNEE*sqrt(1-C4^2),C4);
73  end
74
75  %% TH5
76
77  psi = atan2(sin(TH4)*LL3,cos(TH4)*LL3+LL4);
78
79  ANKLE = 1;
80
81  TH5 = atan2(-ppz,ANKLE*sqrt((ppx+LL5)^2+ppy^2))-psi;
82
83  %% TH6
84
85  TH6 = atan2(ppy,-ppx-LL5);
86
87  if (cos(TH4+TH5)*LL3+cos(TH5)*LL4)<0
88      TH6 = TH6 + pi;
89  end
90
91  %% TH2
92
93  HIP = 1;
94  TH2 = atan2(sin(TH6)*aax+cos(TH6)*aay,HIP*sqrt(1-(sin(TH6)*aax+cos(TH6)*aay)^2));
95
96  %% TH1
97
98  TH1 = atan2(sin(TH6)*ssx+cos(TH6)*ssy,sin(TH6)*nnx+cos(TH6)*nny);
99
100 %% TH3
101
102 TH345 = atan2(aaz,cos(TH6)*aax-sin(TH6)*aay);
103
104 TH3 = TH345 - TH4 - TH5;
105
106 TH3 = TH3 + pi;
```

F.2. Cinemática Brazos

```
1   clear all
2   close all
3   clc
4
5   LA1 = 8.715;
6   LA2 = 10.825; %10.825
7   LA3 = 15.24; %10.24+5cm centro de mano
8
9   t1 = 10;
10  t2 = 10;
11  t3 = 0;
12  t4 = 40;
13
14  t=[t1;
15      t2;
16      t3;
17      t4]
18
19  %Brazo
20  L = 1; %L = 1 derecho L = -1 izquierdo
21
22  %% Cinematica Directa
23
24  A0 = [ 0   0 1   L*LA1;
25         1   0 0     0;
26         0   1 0     0;
27         0   0 0     1];      % B1_0
28
29  A1 = mpaso(t1-90,90,0,0);
30  A2 = mpaso(t2+90,-90,0,0);
31  A3 = mpaso(t3+90,90,0,-LA2);
32  A4 = mpaso(t4-90,180,LA3,0);
33
34  T4_0 = A1*A2*A3*A4;
35
36  TB1_4 = A0*A1*A2*A3*A4;
37
38  %% Cinematica Inversa Geometrica
39
40  %Cinematica Directa
41
```

```
ppx = TB1_4(1,4);
ppy = TB1_4(2,4);
ppz = TB1_4(3,4);

% ppx = 5;
% ppy = 10;
% ppz = -10;

D = [ppy;
     ppz;
     ppx-L*LA1;
     1];

magD = sqrt(D(1)^2+D(2)^2+D(3)^2);

%% TH4 DEF

%Con ley de cosenos

C4 = (LA2^2+LA3^2-magD^2)/(2*LA2*LA3);

if(C4>-1 && C4<1)
    TH4 = atan2(-sqrt(1-C4^2),C4);
else
    TH4 = atan2(0,C4);
end

if TH4<0.00001 && TH4>-0.00001
    TH4 = 0;
end

TH4 = TH4 + pi;
TH4G = TH4*180/pi;
if TH4G>180
    TH4G = TH4G - 360;
end
if TH4G<-180
    TH4G = TH4G + 360;
end
if TH4G<0.00001 && TH4G>-0.00001
    TH4G = 0;
end

%% TH1 DEF

CS = (LA2^2+magD^2-LA3^2)/(2*magD*LA2);
% S = atan2(-1*sqrt(1-CS^2),CS);
S = acos(CS);
SH = S*180/pi

Deq = sqrt(D(1)^2+D(2)^2+0^2);
Scf = Deq/magD;
SHeq = Scf*S;

if((D(2)||D(3))==0)
    TH1 = pi/2 - S;
else
    if(L*D(3)<=0)
        TH1 = pi/2 - atan2(-D(2),D(1))-SHeq;
    else
        TH1 = pi - atan2(D(1),D(2))-S/Scf;
    end
end

TH1G = TH1*180/pi;
if TH1G>180
    TH1G = TH1G - 360;
end
if TH1G<-180
    TH1G = TH1G + 360;
end
if TH1G<0.00001 && TH1G>-0.00001
    TH1G = 0;
end

%% TH2 DEF

if(D(3)==0)
    TH2 = 0;
else
    if(L*D(3)<=0)
        TH2 = 0;
    else
        TH2 = atan2(D(3),D(1));
    end
end

TH2G = TH2*180/pi;
if TH2G>180
    TH2G = TH2G - 360;
end
if TH2G<-180
    TH2G = TH2G + 360;
end
if TH2G<0.00001 && TH2G>-0.00001
    TH2G = 0;
```

```
138 end
139
140 %% TH3  DEF
141 A1 = mpaso(TH1G-90,90,0,0);
142 A2 = mpaso(TH2G+90,-90,0,0);
143
144 T2_0 = A1*A2;
145
146 % D en Ref2
147 D2 = inv(T2_0)*D;
148
149 TH3 = atan2(-D2(1),D2(2));
150
151 if(D(1)<0.00001 && D(1)>-0.00001 && D(2)==L*-24.125 && D(3)<0.00001 && D(3)>-0.00001)
152     TH3 = 0;
153 end
154
155 TH3G = TH3*180/pi;
156 if TH3G>180
157     TH3G = TH3 - 360;
158 end
159 if TH3G<0.00001 && TH3G>-0.00001
160     TH3G = 0;
161 end
```

F.3. Función del Modelo Dinámico

```
1  function [T,Z] = din12(Leg,Q,V,A)
2  L = Leg; % L = 1 RF ,L = -1 LF
3
4  load('dinmat.mat');
5
6  q1r = Q(7,1);
7  q2r = Q(8,1);
8  q3r = Q(9,1);
9  q4r = Q(10,1);
10 q5r = Q(11,1);
11 q6r = Q(12,1);
12
13 q1dr = -V(7,1);
14 q2dr = -V(8,1);
15 q3dr = -V(9,1);
16 q4dr = -V(10,1);
17 q5dr = -V(11,1);
18 q6dr = -V(12,1);
19
20 q1ddr = -A(7,1);
21 q2ddr = -A(8,1);
22 q3ddr = -A(9,1);
23 q4ddr = -A(10,1);
24 q5ddr = -A(11,1);
25 q6ddr = -A(12,1);
26
27 q1l = Q(1,1);
28 q2l = Q(2,1);
29 q3l = Q(3,1);
30 q4l = Q(4,1);
31 q5l = Q(5,1);
32 q6l = Q(6,1);
33
34 q1dl = V(1,1);
35 q2dl = V(2,1);
36 q3dl = V(3,1);
37 q4dl = V(4,1);
38 q5dl = V(5,1);
39 q6dl = V(6,1);
40
41 q1ddl = A(1,1);
42 q2ddl = A(2,1);
43 q3ddl = A(3,1);
44 q4ddl = A(4,1);
45 q5ddl = A(5,1);
46 q6ddl = A(6,1);
47
48 A00 = [1 0 0 L*9.04;
49        0 1 0    0;
50        0 0 1    0;
51        0 0 0    1];
52 A1r = mpaso(q1r,90,0,0);              % LL1 = 4.52 cm
53 A2r = mpaso(q2r-90,-90,0,0);         % LL2 = 5.225 cm
54 A3r = mpaso(q3r,0,9.5659,0); %LL3 = 0.096696 m
55 A4r = mpaso(q4r,0,11.4207,0); %LL4 = 0.114207 m
56 A5r = mpaso(q5r,90,0,0);
57 A6r = mpaso(q6r,0,2.775,0);     %LL5 = 0.02775 m
58
```

```
A1ri = inv(A1r);
A2ri = inv(A2r);
A3ri = inv(A3r);
A4ri = inv(A4r);
A5ri = inv(A5r);
A6ri = inv(A6r);

A1l = A00*mpaso(q1l,90,0,0);              % LL1 = 4.52 cm
A1l = mpaso(q1l,90,0,0);
A2l = mpaso(q2l-90,-90,0,0);        % LL2 = 5.225 cm
A3l = mpaso(q3l,0,9.5659,0); %LL3 = 0.096696 m
A4l = mpaso(q4l,0,11.4207,0); %LL4 = 0.114207 m
A5l = mpaso(q5l,90,0,0);
A6l = mpaso(q6l,0,2.775,0);      %LL5 = 0.02775 m

A1li = inv(A1l);
A2li = inv(A2l);
A3li = inv(A3l);
A4li = inv(A4l);
A5li = inv(A5l);
A6li = inv(A6l);

%b/A b*inv(A)
T5r_6r = A6ri;
T4r_6r = T5r_6r/A5r;
T3r_6r = T4r_6r/A4r;
T2r_6r = T3r_6r/A3r;
T1r_6r = T2r_6r/A2r;
T0r_6r = T1r_6r/A1r;
T1l_6r = T0r_6r*A1l;
T2l_6r = T1l_6r*A2l;
T3l_6r = T2l_6r*A3l;
T4l_6r = T3l_6r*A4l;
T5l_6r = T4l_6r*A5l;
T6l_6r = T5l_6r*A6l;

A1 = mpaso(-q6r,-90,0,0);
A2 = mpaso(-q5r,0,-11.4207,0);
A3 = mpaso(-q4r,0,-9.5659,0);
A4 = mpaso(-q3r+180,-90,0,0)*mpaso(-90,0,0,0);
A5 = mpaso(-q2r,-90,0,0);
A6 = mpaso(-q1r,0,L*9.04,0);
A7 = A1l;
A8 = A2l;
A9 = A3l;
A10 = A4l;
A11 = A5l;
A12 = A6l;

T1_0 = A1;
T2_0 = T1_0*A2;
T3_0 = T2_0*A3;
T4_0 = T3_0*A4;
T5_0 = T4_0*A5;
T6_0 = T5_0*A6;
T7_0 = T6_0*A7;
T8_0 = T7_0*A8;
T9_0 = T8_0*A9;
T10_0 = T9_0*A10;
T11_0 = T10_0*A11;
T12_0 = T11_0*A12;

%% Velocidad Angular

z = [0;
     0;
     1];

w0_0 = [0;
        0;
        0];
w1_1 = A1(1:3,1:3)'*(w0_0+q6dr*z);
w2_2 = A2(1:3,1:3)'*(w1_1+q5dr*z);
w3_3 = A3(1:3,1:3)'*(w2_2+q4dr*z);
w4_4 = A4(1:3,1:3)'*(w3_3+q3dr*z);
w5_5 = A5(1:3,1:3)'*(w4_4+q2dr*z);
w6_6 = A6(1:3,1:3)'*(w5_5+q1dr*z);

w7_7 = A7(1:3,1:3)'*(w6_6+q1dl*z);
w8_8 = A8(1:3,1:3)'*(w7_7+q2dl*z);
w9_9 = A9(1:3,1:3)'*(w8_8+q3dl*z);
w10_10 = A10(1:3,1:3)'*(w9_9+q4dl*z);
w11_11 = A11(1:3,1:3)'*(w10_10+q5dl*z);
w12_12 = A12(1:3,1:3)'*(w11_11+q6dl*z);

%% Aceleracion Angular
%aa^{i}_i = R^{i-1}_i (aa^{i-1}_{i-1} + ro_i qdd_i z + w^{i-1}_{i-1} x ro_i qd_i z )

aa0_0 = [0;
         0;
```

```
          0];

aa1_1 = A1(1:3,1:3)'*(aa0_0+q6ddr*z+cross(w0_0,q6dr*z));
aa2_2 = A2(1:3,1:3)'*(aa1_1+q5ddr*z+cross(w1_1,q5dr*z));
aa3_3 = A3(1:3,1:3)'*(aa2_2+q4ddr*z+cross(w2_2,q4dr*z));
aa4_4 = A4(1:3,1:3)'*(aa3_3+q3ddr*z+cross(w3_3,q3dr*z));
aa5_5 = A5(1:3,1:3)'*(aa4_4+q2ddr*z+cross(w4_4,q2dr*z));
aa6_6 = A6(1:3,1:3)'*(aa5_5+q1ddr*z+cross(w5_5,q1dr*z));

aa7_7 = A7(1:3,1:3)'*(aa6_6+q1ddl*z+cross(w6_6,q1dl*z));
aa8_8 = A8(1:3,1:3)'*(aa7_7+q2ddl*z+cross(w7_7,q2dl*z));
aa9_9 = A9(1:3,1:3)'*(aa8_8+q3ddl*z+cross(w8_8,q3dl*z));
aa10_10 = A10(1:3,1:3)'*(aa9_9+q4ddl*z+cross(w9_9,q4dl*z));
aa11_11 = A11(1:3,1:3)'*(aa10_10+q5ddl*z+cross(w10_10,q5dl*z));
aa12_12 = A12(1:3,1:3)'*(aa11_11+q6ddl*z+cross(w11_11,q6dl*z));

% % Aceleracion Lineal
%al^{i}_i = R^{i-1}_i a^{i-1}_{i-1} + aa^{i}_i x r^{i/i-1}_i + w^{i}_i x (w^{i}_i x r^{i/i-1}_i)

al0_0 = [0;
         0;
         0];

al1_1 =   A1(1:3,1:3)'*al0_0+cross(aa1_1,A1(1:3,1:3)'*A1(1:3,4))+cross(w1_1,cross(w1_1,A1(1:3,1:3)'*A1(1:3,4)));
al2_2 =   A2(1:3,1:3)'*al1_1+cross(aa2_2,A2(1:3,1:3)'*A2(1:3,4))+cross(w2_2,cross(w2_2,A2(1:3,1:3)'*A2(1:3,4)));
al3_3 =   A3(1:3,1:3)'*al2_2+cross(aa3_3,A3(1:3,1:3)'*A3(1:3,4))+cross(w3_3,cross(w3_3,A3(1:3,1:3)'*A3(1:3,4)));
al4_4 =   A4(1:3,1:3)'*al3_3+cross(aa4_4,A4(1:3,1:3)'*A4(1:3,4))+cross(w4_4,cross(w4_4,A4(1:3,1:3)'*A4(1:3,4)));
al5_5 =   A5(1:3,1:3)'*al4_4+cross(aa5_5,A5(1:3,1:3)'*A5(1:3,4))+cross(w5_5,cross(w5_5,A5(1:3,1:3)'*A5(1:3,4)));
al6_6 =   A6(1:3,1:3)'*al5_5+cross(aa6_6,A6(1:3,1:3)'*A6(1:3,4))+cross(w6_6,cross(w6_6,A6(1:3,1:3)'*A6(1:3,4)));

al7_7 =   A7(1:3,1:3)'*al6_6+cross(aa7_7,A7(1:3,1:3)'*A7(1:3,4))+cross(w7_7,cross(w7_7,A7(1:3,1:3)'*A7(1:3,4)));
al8_8 =   A8(1:3,1:3)'*al7_7+cross(aa8_8,A8(1:3,1:3)'*A8(1:3,4))+cross(w8_8,cross(w8_8,A8(1:3,1:3)'*A8(1:3,4)));
al9_9 =   A9(1:3,1:3)'*al8_8+cross(aa9_9,A9(1:3,1:3)'*A9(1:3,4))+cross(w9_9,cross(w9_9,A9(1:3,1:3)'*A9(1:3,4)));
al10_10 = A10(1:3,1:3)'*al9_9+cross(aa10_10,A10(1:3,1:3)'*A10(1:3,4))+cross(w10_10,cross(w10_10,A10(1:3,1:3)'*A10(1:3,4)));
al11_11 = A11(1:3,1:3)'*al10_10+cross(aa11_11,A11(1:3,1:3)'*A11(1:3,4))+cross(w11_11,cross(w11_11,A11(1:3,1:3)'*A11(1:3,4)))
    ;
al12_12 = A12(1:3,1:3)'*al11_11+cross(aa12_12,A12(1:3,1:3)'*A12(1:3,4))+cross(w12_12,cross(w12_12,A12(1:3,1:3)'*A12(1:3,4)))
    ;

% % Aceleracion del centro de masa
%a^G_i = a^i_i + aa^i_i x r^{G/i}_i + w^i_i x ( w^i_i x r^{G/i}_i)

ag1 = al1_1+cross(aa1_1,FJacopladosTrOG_A)+cross(w1_1,cross(w1_1,FJacopladosTrOG_A));
ag2 = al2_2+cross(aa2_2,FJrtrOG_A)+cross(w2_2,cross(w2_2,FJrtrOG_A));
ag3 = al3_3+cross(aa3_3,FRodillarOG_A)+cross(w3_3,cross(w3_3,FRodillarOG_A));
ag4 = al4_4+cross(aa4_4,FJacopladosCrOG_A)+cross(w4_4,cross(w4_4,FJacopladosCrOG_A));
ag5 = al5_5+cross(aa5_5,FCaderarOG_A)+cross(w5_5,cross(w5_5,FCaderarOG_A));
ag6 = al6_6+cross(aa6_6,FJCadrOG_A)+cross(w6_6,cross(w6_6,FJCadrOG_A));

ag7 = al7_7+cross(aa7_7,CaderarOG_A)+cross(w7_7,cross(w7_7,CaderarOG_A));
ag8 = al8_8+cross(aa8_8,JacopladosCrOG_A)+cross(w8_8,cross(w8_8,JacopladosCrOG_A));
ag9 = al9_9+cross(aa9_9,RodillarOG_A)+cross(w9_9,cross(w9_9,RodillarOG_A));
ag10 = al10_10+cross(aa10_10,JrtrOG_A)+cross(w10_10,cross(w10_10,JrtrOG_A));
ag11 = al11_11+cross(aa11_11,JacopladosTrOG_A)+cross(w11_11,cross(w11_11,JacopladosTrOG_A));
ag12 = al12_12+cross(aa12_12,PierOG_A)+cross(w12_12,cross(w12_12,PierOG_A));

% % Fuerza inercial de cada eslabon

F1 = Jacopladosm*ag1;
F2 = Jrtm*ag2;
F3 = Rodillam*ag3;
F4 = JacopladosCm*ag4;
F5 = Caderam*ag5;
F6 = JCadm*ag6;
F7 = Caderam*ag7;
F8 = JacopladosCm*ag8;
F9 = Rodillam*ag9;
F10 = Jrtm*ag10;
F11 = Jacopladosm*ag11;
F12 = Piem*ag12;

% % Momento inercial de cada eslabon

M1 = FJacopladosTI*aa1_1+cross(w1_1,FJacopladosTI*w1_1);
M2 = FJrtI*aa2_2+cross(w2_2,FJrtI*w2_2);
M3 = FRodillaI*aa3_3+cross(w3_3,FRodillaI*w3_3);
M4 = FJacopladosCI*aa4_4+cross(w4_4,FJacopladosCI*w4_4);
M5 = FCaderaI*aa5_5+cross(w5_5,FCaderaI*w5_5);
M6 = FJCadI*aa6_6+cross(w6_6,FJCadI*w6_6);
M7 = CaderaI*aa7_7+cross(w7_7,CaderaI*w7_7);
M8 = JacopladosCI*aa8_8+cross(w8_8,JacopladosCI*w8_8);
M9 = RodillaI*aa9_9+cross(w9_9,RodillaI*w9_9);
M10 = JrtI*aa10_10+cross(w10_10,JrtI*w10_10);
M11 = JacopladosTI*aa11_11+cross(w11_11,JacopladosTI*w11_11);
M12 = PieI*aa12_12+cross(w12_12,PieI*w12_12);

% % Propagacion hacia atras de Newton
%f^{i,i-1}_i = f^{i+1,i}_i - m_i R^0_i g_0 + F_i
%f^{i,i-1}_{i-1} R^{i}_{i-1} f^{i,i-1}_i
```

```
g0 = [981; 0;  0];
f131213 = [0;
           0;
           0];
f131212 = [0;
           0;
           0];
f121112 = f131212 - Piem*T12_0(1:3,1:3)'*g0+F12;
f111011 = A12(1:3,1:3)*f121112 - Jacopladosm*T11_0(1:3,1:3)'*g0+F11;
f10910 = A11(1:3,1:3)*f111011 - Jrtm*T10_0(1:3,1:3)'*g0+F10;
f989 = A10(1:3,1:3)*f10910 - Rodillam*T9_0(1:3,1:3)'*g0+F9;
f878 = A9(1:3,1:3)*f989 - JacopladosCm*T8_0(1:3,1:3)'*g0+F8;
f767 = A8(1:3,1:3)*f878 - Caderam*T7_0(1:3,1:3)'*g0+F7;
f656 = A7(1:3,1:3)*f767 - JCadm*T6_0(1:3,1:3)'*g0+F6;
f545 = A6(1:3,1:3)*f656 - Caderam*T5_0(1:3,1:3)'*g0+F5;
f434 = A5(1:3,1:3)*f545 - JacopladosCm*T4_0(1:3,1:3)'*g0+F4;
f323 = A4(1:3,1:3)*f434 - Rodillam*T3_0(1:3,1:3)'*g0+F3;
f212 = A3(1:3,1:3)*f323 - Jrtm*T2_0(1:3,1:3)'*g0+F2;
f101 = A2(1:3,1:3)*f212 - Jacopladosm*T1_0(1:3,1:3)'*g0+F1;
f100 = A1(1:3,1:3)*f101;

%% Propagacion hacia atras de Euler
%n^{i,i-1}_i = n^{i+1,i}_i + (r^{i/i-1}_i + r^{G/i}_i)x f^{i,i-1}_i - r^{G/i}_i x f^{i+1,i}_i + MG_i

n131212 = [0;
           0;
           0];

n121112 = n131212 +              cross((A12(1:3,1:3)'*A12(1:3,4)+PierOG_A),f121112) -        cross(PierOG_A,f131212) +
      M12;
n111011 = A12(1:3,1:3)*n121112 + cross((A11(1:3,1:3)'*A11(1:3,4)+JacopladosTrOG_A),f111011) - cross(JacopladosTrOG_A,
      A12(1:3,1:3)*f121112) + M11;
n10910 =  A11(1:3,1:3)*n111011 + cross((A10(1:3,1:3)'*A10(1:3,4)+JrtrOG_A),f10910) -        cross(JrtrOG_A,         A11
      (1:3,1:3)*f111011) + M10;
n989 =    A10(1:3,1:3)*n10910 +  cross((A9(1:3,1:3)'*A9(1:3,4)+RodillarOG_A),f989) -        cross(RodillarOG_A,     A10
      (1:3,1:3)*f10910) + M9;
n878 =    A9(1:3,1:3)*n989 +     cross((A8(1:3,1:3)'*A8(1:3,4)+JacopladosCrOG_A),f878) -     cross(JacopladosCrOG_A, A9
      (1:3,1:3)*f989) + M8;
n767 =    A8(1:3,1:3)*n878 +     cross((A7(1:3,1:3)'*A7(1:3,4)+CaderarOG_A),f767) -          cross(CaderarOG_A,      A8
      (1:3,1:3)*f878) + M7;
n656 =    A7(1:3,1:3)*n767 +     cross((A6(1:3,1:3)'*A6(1:3,4)+FJCadrOG_A),f656) -          cross(FJCadrOG_A,       A7
      (1:3,1:3)*f767) + M6;
n545 =    A6(1:3,1:3)*n656 +     cross((A5(1:3,1:3)'*A5(1:3,4)+FCaderarOG_A),f545) -        cross(FCaderarOG_A,     A6
      (1:3,1:3)*f656) + M5;
n434 =    A5(1:3,1:3)*n545 +     cross((A4(1:3,1:3)'*A4(1:3,4)+FJacopladosCrOG_A),f434) -    cross(FJacopladosCrOG_A,A5
      (1:3,1:3)*f545) + M4;
n323 =    A4(1:3,1:3)*n434 +     cross((A3(1:3,1:3)'*A3(1:3,4)+FRodillarOG_A),f323) -       cross(FRodillarOG_A,    A4
      (1:3,1:3)*f434) + M3;
n212 =    A3(1:3,1:3)*n323 +     cross((A2(1:3,1:3)'*A2(1:3,4)+FJrtrOG_A),f212) -           cross(FJrtrOG_A,        A3(1:3,1:3)
      *f323) + M2;
n101 =    A2(1:3,1:3)*n212 +     cross((A1(1:3,1:3)'*A1(1:3,4)+FJacopladosTrOG_A),f101) -    cross(FJacopladosTrOG_A,A2
      (1:3,1:3)*f212) + M1;

%% Torques
% T_i = n^{i,i-1}_i-1 . z

T6l = z'*A12(1:3,1:3)*n121112;
T5l = z'*A11(1:3,1:3)*n111011;
T4l = z'*A10(1:3,1:3)*n10910;
T3l = z'*A9(1:3,1:3)*n989;
T2l = z'*A8(1:3,1:3)*n878;
T1l = z'*A7(1:3,1:3)*n767;
T1r = z'*A6(1:3,1:3)*n656;
T2r = z'*A5(1:3,1:3)*n545;
T3r = z'*A4(1:3,1:3)*n434;
T4r = z'*A3(1:3,1:3)*n323;
T5r = z'*A2(1:3,1:3)*n212;
T6r = z'*A1(1:3,1:3)*n101;


T = zeros(12,1);
T(1,1) = T1l/10000000;
T(2,1) = T2l/10000000;
T(3,1) = T3l/10000000;
T(4,1) = T4l/10000000;
T(5,1) = T5l/10000000;
T(6,1) = T6l/10000000;
T(7,1) = T1r/10000000;
T(8,1) = T2r/10000000;
T(9,1) = T3r/10000000;
T(10,1) = T4r/10000000;
T(11,1) = T5r/10000000;
T(12,1) = T6r/10000000;

%T/10000000 %g cm / s2 * cm a kg cm cm
Z = zmp(f100/100000,(A1(1:3,1:3)*n101)/10000000);

end
```

Apéndice G

Código C

G.1. Función de Caminado Parametrizado

```
1  double L_SCI(double t, double q[12], int flags [5], double Desp[6]) //f[] 0=RF,1=ST,2=EN,3=pa,4=np
2  {
3      int L,i;
4      double lds = LL1;
5      double XB=0,YB=0,ZB=0,XF=0,YF=0,ZF=0;
6      double M1 [4][4]={{1,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1}};
7      double M2 [4][4]={{1,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1}};
8      double q1[6]={0,0,0,0,0,0};
9      double q2[6]={0,0,0,0,0,0};
10     double h=height;
11     if (flags[RF]==1) //RF
12     {
13         L = 1;
14     }
15     else
16     {
17         L = -1;
18     }
19
20     /*% Rafa      YO
21     % X    =    Y
22     % Y    =    Z
23     % Z    =    X
24     %SCI(t,Ti,x0,x1,v0,v1)*/
25
26     if(flags[ST] == 1 && flags[EN] == 0) //START END
27         {
28             if(t<=Tds/2.0)
29             {
30                 YB = SCI(t,Tp,0,Lp/2.0,0,0);
31                 ZB = SCI(t,Tp/2.0,h,h+hb,0,0);
32                 XB = SCI(t,Tp/2.0,-L*lds,L*db,0,0);
33
34                 YF = 0;
35                 ZF = 0;
36                 XF = -2.0*L*lds;
37
38
39             }
40             if(t>Tds/2.0 && t<=Tp/2.0)
41             {
42                 YB = SCI(t,Tp,0,Lp/2.0,0,0);
43                 ZB = SCI(t,Tp/2.0,h,h+hb,0,0);
44                 XB = SCI(t,Tp/2.0,-L*lds,L*db,0,0);
45
46                 YF = SCI(t-Tds/2.0,Tp-Tds,0,Lp,0,0);
47                 ZF = SCI(t-Tds/2.0,(Tp-Tds)/2.0,0,hp,0,0);
48                 XF = SCI(t-Tds/2.0,(Tp-Tds)/2.0,-2.0*L*lds,-L*dp,0,0);
49             }
50             if(t>Tp/2.0 && t<=Tp-Tds/2.0)
51             {
52                 YB = SCI(t,Tp,0,Lp/2.0,0,0);
53                 ZB = SCI(t-Tp/2.0,Tp/2.0,h+hb,h,0,0);
54                 XB = SCI(t-Tp/2.0,Tp/2.0,L*db,-L*lds,0,0);
55
56                 YF = SCI(t-Tds/2.0,Tp-Tds,0,Lp,0,0);
57                 ZF = SCI(t-Tp/2.0,(Tp-Tds)/2.0,hp,0,0,0);
58                 XF = SCI(t-Tp/2.0,(Tp-Tds)/2.0,-L*dp,-2.0*L*lds,0,0);
59             }
60             if(t>Tp-Tds/2.0 && t<=Tp)
```

```
61              {
62                  YB = SCI(t,Tp,0,Lp/2.0,0,0);
63                  ZB = SCI(t-Tp/2.0,Tp/2.0,h+hb,h,0,0);
64                  XB = SCI(t-Tp/2.0,Tp/2.0,L*db,-L*lds,0,0);
65
66                  YF = Lp;
67                  ZF = 0;
68                  XF = -2.0*L*lds;
69              }
70
71          }
72      if(flags[ST] == 0 && flags[EN] == 0)
73      {
74          if(t<=Tds/2.0)
75          {
76              YB = SCI(t,Tp,-Lp/2.0,Lp/2.0,0,0);
77              ZB = SCI(t,Tp/2.0,h,h+hb,0,0);
78              XB = SCI(t,Tp/2.0,-L*lds,L*db,0,0);
79
80              YF = -Lp;
81              ZF = 0;
82              XF = -2.0*L*lds;
83          }
84          if(t>Tds/2.0 && t<=Tp/2.0)
85          {
86              YB = SCI(t,Tp,-Lp/2.0,Lp/2.0,0,0);
87              ZB = SCI(t,Tp/2.0,h,h+hb,0,0);
88              XB = SCI(t,Tp/2.0,-L*lds,L*db,0,0);
89
90              YF = SCI(t-Tds/2.0,Tp-Tds,-Lp,Lp,0,0);
91              ZF = SCI(t-Tds/2.0,(Tp-Tds)/2.0,0,hp,0,0);
92              XF = SCI(t-Tds/2.0,(Tp-Tds)/2.0,-2.0*L*lds,-L*dp,0,0);
93          }
94          if(t>Tp/2.0 && t<=Tp-Tds/2.0)
95          {
96              YB = SCI(t,Tp,-Lp/2.0,Lp/2.0,0,0);
97              ZB = SCI(t-Tp/2.0,Tp/2.0,h+hb,h,0,0);
98              XB = SCI(t-Tp/2.0,Tp/2.0,L*db,-L*lds,0,0);
99
100             YF = SCI(t-Tds/2.0,Tp-Tds,-Lp,Lp,0,0);
101             ZF = SCI(t-Tp/2.0,(Tp-Tds)/2.0,hp,0,0,0);
102             XF = SCI(t-Tp/2.0,(Tp-Tds)/2.0,-L*dp,-2.0*L*lds,0,0);
103         }
104         if(t>Tp-Tds/2.0 && t<=Tp)
105         {
106             YB = SCI(t,Tp,-Lp/2.0,Lp/2.0,0,0);
107             ZB = SCI(t-Tp/2.0,Tp/2.0,h+hb,h,0,0);
108             XB = SCI(t-Tp/2.0,Tp/2.0,L*db,-L*lds,0,0);
109
110             YF = Lp;
111             ZF = 0;
112             XF = -2.0*L*lds;
113         }
114     }
115     if(flags[ST] == 0 && flags[EN] == 1)
116     {
117         if(t<=Tds/2.0)
118         {
119             YB = SCI(t,Tp,-Lp/2.0,0,0,0);
120             ZB = SCI(t,Tp/2.0,h,h+hb,0,0);
121             XB = SCI(t,Tp/2.0,-L*lds,L*db,0,0);
122
123             YF = -Lp;
124             ZF = 0;
125             XF = -2.0*L*lds;
126         }
127         if(t>Tds/2.0 && t<=Tp/2.0)
128         {
129             YB = SCI(t,Tp,-Lp/2.0,0,0,0);
130             ZB = SCI(t,Tp/2.0,h,h+hb,0,0);
131             XB = SCI(t,Tp/2.0,-L*lds,L*db,0,0);
132
133             YF = SCI(t-Tds/2.0,Tp-Tds,-Lp,0,0,0);
134             ZF = SCI(t-Tds/2.0,(Tp-Tds)/2.0,0,hp,0,0);
135             XF = SCI(t-Tds/2.0,(Tp-Tds)/2.0,-2.0*L*lds,-L*dp,0,0);
136         }
137         if(t>Tp/2.0 && t<=Tp-Tds/2.0)
138         {
139             YB = SCI(t,Tp,-Lp/2.0,0,0,0);
140             ZB = SCI(t-Tp/2.0,Tp/2.0,h+hb,h,0,0);
141             XB = SCI(t-Tp/2.0,Tp/2.0,L*db,-L*lds,0,0);
142
143             YF = SCI(t-Tds/2.0,Tp-Tds,-Lp,0,0,0);
144             ZF = SCI(t-Tp/2.0,(Tp-Tds)/2.0,hp,0,0,0);
145             XF = SCI(t-Tp/2.0,(Tp-Tds)/2.0,-L*dp,-2.0*L*lds,0,0);
146         }
147         if(t>Tp-Tds/2.0 && t<=Tp)
148         {
149             YB = SCI(t,Tp,-Lp/2.0,0,0,0);
150             ZB = SCI(t-Tp/2.0,Tp/2.0,h+hb,h,0,0);
```

```
151             XB = SCI(t-Tp/2.0,Tp/2.0,L*db,-L*lds,0,0);
152
153             YF = 0;
154             ZF = 0;
155             XF = -2.0*L*lds;
156         }
157     }
158     t+=deltaT;
159     if(t>Tp)
160     {
161         if(flags[RF]==1) //RF
162             flags[RF] = 0;
163         else
164             flags[RF] = 1;
165         t = deltaT;
166         flags[PA]=flags[PA]+1; //paso
167         if(flags[ST] == 1 && flags[EN] == 0)
168             flags[ST] = 0;
169         if(flags[PA]>=flags[NP]) //paso>=numpasos
170             flags[EN] = 1;
171     }
172
173     M1[0][3] = XB;
174     M1[1][3] = YB;
175     M1[2][3] = ZB;
176
177     M2[0][3] = XF-XB;
178     M2[1][3] = YF-YB;
179     M2[2][3] = ZF-ZB;
180
181     IKBases(0,L,M1,q1);
182     IKBases(1,-L,M2,q2);
183
184     if (L==-1)
185     {
186         for (i=0;i<6;i++)
187         {
188             q[i]=q1[i];
189         }
190         for (i=0;i<6;i++)
191         {
192             q[i+6]=q2[i];
193         }
194     }
195     else
196     {
197         for (i=0;i<6;i++)
198         {
199             q[i]=q2[i];
200         }
201         for (i=0;i<6;i++)
202         {
203             q[i+6]=q1[i];
204         }
205     }
206
207     Desp[0] = XB;
208     Desp[1] = YB;
209     Desp[2] = ZB;
210     Desp[3] = XF;
211     Desp[4] = YF;
212     Desp[5] = ZF;
213
214     return t;
215 }
```

G.2. Función de Caminado Omnidireccional

```
1  void omni(double q[6], double VRobot[3], double psi, int ls, double pLeg, int cont, int col)
2  {
3    double vxRobot = VRobot[0];
4    double vyRobot = VRobot[1];
5    double vtRobot = VRobot[2];
6    double aRobot[3];
7    double arRobot,apRobot,ayRobot;
8
9    arRobot = asin(vxRobot/(psi*height));
10   apRobot = asin(vyRobot/(psi*height));
11   ayRobot = vtRobot/psi;
12
13   //SHIFTING
14   double aShift,tShift,tLegShift,tFootShift;
15   if(arRobot>=0)
16     aShift = 0.12+0.08*sqrt(arRobot*arRobot+apRobot*apRobot)+0.7*arRobot; //%######## 0.12+0.08*sqrt(arRobot*
       arRobot+apRobot*apRobot)+0.7*arRobot;
```

```
17      else
18        aShift = 0.12+0.08*sqrt(arRobot*arRobot+apRobot*apRobot)+0.7*-arRobot; //%######## 0.12+0.08*sqrt(arRobot*
            arRobot+apRobot*apRobot)+0.7*-arRobot;
19
20      tShift  = aShift*sin(pLeg);
21      tLegShift = tShift;
22      tFootShift = -0.2*tShift;
23
24      //SHORTENING
25      double pShort,aShort,gShort,tFootShort;
26      pShort = vShort*(pLeg+PI/2.0+oShort);
27      aShort = 0.2 + 2.0*sqrt(arRobot*arRobot+apRobot*apRobot);
28
29      if(pShort>=-PI && pShort<PI)
30      {
31        gShort = -aShort*0.5*(cos(pShort)+1.0); //%################### -aShort*0.5*(cos(pShort)+1);
32        tFootShort = -apRobot*0.125*(cos(pShort)+1.0); //%################## art -apRobot*0.125*(cos(pShort)+1);
            mod apRobot*0.5*(cos(pShort)+1);
33      }
34      else
35      {
36        gShort = 0;
37        tFootShort = 0;
38      }
39
40
41      //LOADING
42      double in = pLeg+PI/2.0-PI/vShort+oShort,out,pLoad,aLoad,gLoad;
43      while (in<-PI || in>=PI)
44      {
45        out = piCut(in);
46        in = out;
47      }
48      pLoad = vLoad*in-PI;
49
50      if(apRobot>=0)
51        aLoad = 0.025+0.5*(1.0-cos(apRobot)); //%##########
52      else
53        aLoad = 0.025+0.5*(1.0-cos(-apRobot)); //%##########
54
55      if(pLoad>=-PI && pLoad<PI)
56        gLoad = -aLoad*0.5*(cos(pLoad)+1.0);
57      else
58        gLoad = 0;
59
60      //SWINGING
61      double pSwing,b,tSwing,trLS,tpLS,tyLS,trFS,tpFS;
62
63      pSwing = vSwing*(pLeg+PI/2.0+oSwing);
64      b = -2.0/(2.0*PI*vSwing-PI); //%Speed of reverse motion ###########
65
66      if(pSwing>=-PI/2.0 && pSwing<PI/2.0)
67        tSwing = sin(pSwing);
68      else if(pSwing>=PI/2.0)
69        tSwing = b*(pSwing-PI/2.0)+1.0;  //%##############
70      else
71        tSwing = b*(pSwing+PI/2.0)-1.0;  //%##############
72
73      trLS = ls*arRobot*tSwing;
74      tpLS = apRobot*tSwing;
75      tyLS = ls*ayRobot*tSwing;
76      trFS = ls*0.25*arRobot*tSwing; //%##############
77      tpFS = 0.25*apRobot*tSwing; //%############## 0.25*apRobot*tSwing;
78
79      //BALANCE
80      double trFB,tpFB,trLB;
81
82      trFB = 0.1*ls*arRobot*cos(pLeg+0.35); //%################## 0.1*ls*arRobot*cos(pLeg+0.35);
83      tpFB = 0.02+0.08*apRobot-0.04*apRobot*cos(2.0*pLeg+0.7); //%######### 0.02+0.08*apRobot-0.04*apRobot*cos
            (2*pLeg+0.7);
84      if(arRobot>=0)
85       trLB = 0.01+ls*arRobot+arRobot+0.1*ayRobot; //%0.01+ls*arRobot+arRobot+0.1*ayRobot; 0.05 no side collision
86      else
87       trLB = 0.01+ls*arRobot-arRobot+0.1*ayRobot; //%0.01+ls*arRobot-arRobot+0.1*ayRobot; 0.05 no side collision
88
89      //OUTPUT
90      double trLeg,tpLeg,tyLeg,trFoot,tpFoot,tFoot,g;
91      trLeg = trLS+tLegShift+trLB; //%----------
92      tpLeg = tpLS;
93      tyLeg = tyLS;
94      trFoot = trFS+tFootShift+trFB; //%---------
95      tpFoot = tpFS+tFootShort+tpFB;
96      //tFoot = [trFoot;tpFoot];
97      g = gShort+gLoad;
98
99      //LEG INTERFACE
100     //% -1<gamma<0, 0 fully extended -1 nmin
101
102     double n = 1+(1-nmin)*g; //% Target relative leg length
103     double tKnee,tThigh,tHip[2],tAnkle[2],TH1,TH2,TH3,TH4,TH5,TH6;
104
105     tKnee = -2.0*acos(n);
106     tThigh = tyLeg;
```

```
107
108     if(tyLeg==0)
109     {
110       //tHip = [trLeg;tpLeg]+[0;-0.5*tKnee]; %0.5 same size thigh and shank
111       tHip[0] = trLeg;
112       tHip[1] = tpLeg-0.5*tKnee;
113       //tAnkle = [0;-0.5*tKnee]+tFoot-[trLeg;tpLeg]; %0.5 same size thigh and shank
114       tAnkle[0] = 0+trFoot-trLeg;
115       tAnkle[1] = -0.5*tKnee+tpFoot-tpLeg;
116     }
117     else
118     {
119       tHip[0] = trLeg;
120       tHip[1] = tpLeg-0.5*tKnee;
121       tAnkle[0] = 0+trFoot-trLeg;
122       tAnkle[1] = -0.5*tKnee+tpFoot-tpLeg;
123     }
124
125     TH1 = -ls*tThigh;
126     TH2 = ls*tHip[0]; //%##################################
127     TH3 = -tHip[1];
128     TH4 = -tKnee;
129     TH5 = -tAnkle[1];
130     TH6 = ls*tAnkle[0]; //%##################################
```

G.3. Seguimiento y alcance de objetos

```
1       while(1)
2       {
3       cvGrabFrame(cap1);
4       cvGrabFrame(cap1);
5       cvGrabFrame(cap1);
6       cvGrabFrame(cap1);
7       cvGrabFrame(cap2);
8       cvGrabFrame(cap2);
9       cvGrabFrame(cap2);
10      cvGrabFrame(cap2);
11      right = cvQueryFrame(cap1); // get frame
12          left = cvQueryFrame(cap2); // get frame
13          if(!right)
14          {
15            printf("Capture is null");
16          }
17
18      cvRemap(left, left, mx1, my1, CV_INTER_LINEAR, cvScalarAll(0));
19      cvRemap(right, right, mx2, my2, CV_INTER_LINEAR, cvScalarAll(0));
20
21
22
23          hsvr = cvCreateImage( cvGetSize(right), right->depth, 3);
24      hsvl = cvCreateImage( cvGetSize(left), left->depth, 3);
25
26
27      segm(&right,&hsvr,&h,&s,&v,&ss,&s2,&segr,0);
28
29  //
30      storageR = cvCreateMemStorage(0);
31          //circlesR = cvHoughCircles(dstdu, storageR, CV_HOUGH_GRADIENT, 2,cvGetSize(img).height/2,param1,
          param2, minradius, maxradius);
32      circlesR = cvHoughCircles(segr, storageR, CV_HOUGH_GRADIENT, 2,cvGetSize(right).height/2,300,42, 20, 100);
33          for(i = 0; i < circlesR->total; i++) {
34      pR = (float*) cvGetSeqElem( circlesR, i );
35      ptR = cvPoint( cvRound( pR[0] ), cvRound( pR[1] ) );
36      cvCircle(right,ptR,cvRound( pR[2] ),CV_RGB(0xff,0xff,0xff),1,8,2);
37
38          //printf("Centroid Right (x,y)=(%f,%f) \n",pR[0],pR[1]);
39          xr=cvRound(pR[0]);
40              yr=cvRound(pR[1]);
41
42      }
43
44  //
45      cvShowImage("Video_R", segr); // show frame
46      cvShowImage("Video_RH", right); // show frame
47
48
49
50      segm(&left,&hsvl,&h,&s,&v,&ss,&s2,&segl,1);
51
52  //
53      storageR = cvCreateMemStorage(0);
54          //circlesR = cvHoughCircles(dstdu, storageR, CV_HOUGH_GRADIENT, 2,cvGetSize(img).height/2,param1,
          param2, minradius, maxradius);
55      circlesR = cvHoughCircles(segl, storageR, CV_HOUGH_GRADIENT, 2,cvGetSize(left).height/2,300,42, 20, 100);
```

```
56          for(i = 0; i < circlesR->total; i++) {
57      pR = (float*) cvGetSeqElem( circlesR, i );
58      ptR = cvPoint( cvRound( pR[0] ), cvRound( pR[1] ) );
59      cvCircle(left ,ptR,cvRound( pR[2] ),CV_RGB(0xff,0xff,0xff),1,8,2);
60
61          //printf("Centroid L(x,y)=(%f, %f) \n",pR[0],pR[1]);
62          xl=cvRound(pR[0]);
63              yl=cvRound(pR[1]);
64      }
65
66  //
67
68          cvShowImage( "Video L", segl); // show frame
69      cvShowImage("Video LH", left); // show frame
70
71
72  //
73      //centroid(segr, &xr, &yr);
74      //centroid(segl, &xl, &yl);
75  //
76          printf("Centroid R(x,y)=(%d,%d) \n",xr,yr);
77      printf("Centroid L(x,y)=(%d,%d) \n",xl,yl);
78
79
80      if(xr>xl)
81        d = xr-xl;
82      else
83        d = xl-xr;
84
85      X = xl * q00 + q03;
86      Y = yl * q11 + q13;
87      Z = q23;
88      W = d * q32 + q33;
89
90      X = X / W;
91      Y = Y / W;
92      Z = Z / W;
93
94      X = X-3.0;
95
96      printf("X = %f \n Y = %f \n Z = %f \n",X,Y,Z);
97
98      pan =  atan2(-X,Z); //pdh -(X-3)
99      tilt  = atan2(-Y,Z); //tdh
100
101     tilt  = tilt*(180.0/PI);
102     pan = pan*(180.0/PI);
103
104
105     if( tilt >-20 && tilt<20)
106     {
107       tilt1  = tilt;
108       if( tilt1 <1 && tilt1>0)
109         tilt1 =tilt1+1;
110       if( tilt1 <0 && tilt1>-1)
111         tilt1 =tilt1-1;
112     }
113     if(pan>-20 && pan<20)
114     {
115       pan1 = pan;
116       if(pan1<1 && pan1>0)
117         pan1=pan1+1;
118       if(pan1<0 && pan1>-1)
119         pan1=pan1-1;
120     }
121
122     pf = pan1-pana;
123     if(pf<0)
124       pf=pf*(-1.0);
125     tf = tilt1-tilta;
126     if(tf<0)
127       tf=tf*(-1.0);
128
129      if( pf>2 || tf>2)
130       {
131         new = 1;
132       }
133       else
134       {
135         new = 0;
136       }
137
138
139     printf(" tilt1 = %.2f \n pan1 = %.2f \n",tilt1,pan1);
140
141     if(X<10 && X>-10 && Z<40 && Z>1)
142     {
143       //tiltabs = tiltabs+tilt1;
144       //panabs = panabs+pan1;
145       p = panabs*(PI/180.0);
146       t = tiltabs*(PI/180.0);
147
148       x = X*cos(p)+(LH2-Y)*sin(p)*sin(t)-(Z+LH3)*sin(p)*cos(t);
```

```
149         y = X*sin(p)+(Y-LH2)*cos(p)*sin(t)+(Z+LH3)*cos(p)*cos(t);
150         z = (LH2-Y)*cos(t)+(Z+LH3)*sin(t)+LH1;
151
152         printf("tabs = %f pabs = %f x= %f y= %f z= %f\n",tiltabs,panabs,x,y,z);
153         magD = sqrt((x+LA1)*(x+LA1)+y*y+z*z);
154
155         if(magD<(LA2+LA3))
156         {
157           printf("mag <");
158           pos[0] = x;
159           pos[1] = y;
160           pos[2] = z;
161           IKBrazos(-1, pos, q);
162         }
163         else
164         {
165           printf("mag >");
166           q[0]=0;
167           q[1]=0;
168           q[2]=0;
169           q[3]=0;
170         }
171         if(q[0]<1&&q[0]>-1)
172           q[0]=q[0]+1.1;
173         if(q[1]<1&&q[1]>-1)
174           q[1]=q[1]+1.1;
175         if(q[2]<1&&q[2]>-1)
176           q[2]=q[2]+1.1;
177             if(q[3]<1&&q[3]>-1)
178           q[3]=q[3]+1.1;
179         for (i=0;i<4;i++)
180         {
181           printf("TH%d = %f \n",i+1,q[i]);
182         }
183         //tiltabs = tiltabs-tilt1;
184         //panabs = panabs-pan1;
185         q[4] = 1.1;
186         q[5] = 24.9;
187
188       //c = cvWaitKey(0);
189       c = 99;
190       //printf("c = %d\n",c);
191       if(c == 99) //if(c==c)  //49=1 51=3
192
193
194       if(( tilt1 >3 ||  tilt1 <-3 || pan1>3 || pan1<-3) && new)
195       {
196           tiltabs = tiltabs+tilt1;
197           panabs = panabs+pan1;
198           printf(" tilta = %.2f \n pana = %.2f \n",tiltabs,panabs);
199           pana = pan1;
200           tilta = tilt1;
201           head(data,data1,data2,data3,data4,&tiltabs,&panabs,&abp,&aby);
202           tx(&fp,data);
203       }
204       else
205       {
206           if(magD<(LA2+LA3))
207           {
208             printf("\nDentro de espacio\n");
209             larm(data,data1,data2,data3,data4,data5,data6,q);
210             tx(&fp,data);
211             stop = 1;
212             waitFor(5);
213             //q[5]=19.9;
214             //q[5]=15.9;
215             q[5]=11.9;
216             larm(data,data1,data2,data3,data4,data5,data6,q);
217             tx(&fp,data);
218             waitFor(2);
219             q[0]=1.1;
220             q[1]=1.1;
221             q[2]=1.1;
222             q[3]=1.1;
223             q[4]=1.1;
224             //q[5]=19.9;
225             q[5]=11.9;
226             larm(data,data1,data2,data3,data4,data5,data6,q);
227             tx(&fp,data);
228           }
229           else
230           {
231             printf("Fuera del espacio");
232           }
233       }
234
235       }
236
237           c = cvWaitKey(10); // wait 10 ms or for key stroke
238           //c = cvWaitKey(0); // wait for key stroke
```

```
239             if(c == 27)
240                 break; // if ESC, break and quit
241         if(stop)
242             break;
243         }
```

Bibliografía

[1] International Organization for Standardization. ISO 8373:2012. https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en, 2012.

[2] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer, 2008.

[3] Jens Christensen, Jesper L. Nielsen, Mads S. Svendsen, Mikael S. Svenstrup, Kasper Winther, and Peter F. Orts. Modelling and control of a biped robot. Master's thesis, Aalborg University, 2006.

[4] Mikkel Melters Pedersen, Allan Agerbo Nielsen, and Lars Fuglsang Christiansen. Design of biped robot aau-bot1. Master's thesis, Aalborg University, 2007.

[5] Jens Christensen, Jesper L. Nielsen, Mads S. Svendsen, and Peter F. Ortsn. Development, modeling and control of a humanoid robot. Master's thesis, Aalborg University, 2007.

[6] Karl Muecke, Jess Kanetzky, Raghav Sampath, and Patrick Cox. Darwin: a bipedal walking humanoid robot. http://web.stevens.edu/msrobotics/SMRDC2010/muecke06r.pdf, 2006.

[7] Jason Falconer. Dr. guero's back with another robot balancing act. http://www.gizmag.com/dr-guero-balancing-robot/26860/, 2013.

[8] Dr. Guero. Biped robot walks just like a human being. http://ai2001.ifdef.jp/, 2013.

[9] Yoon, Sukjune, Seungyong Hyung, Minhyung Lee, Kyung Shik Roh, SungHwan Ahn, Andrew Gee, Pished Bunnun, Andrew Calway, and Waterio W. Mayol Cue-

vas. Humanoid robot that sees and maps. http://www.bris.ac.uk/news/2013/9536.htm, 2013.

[10] Robocup. http://www.robocup2014.org/, 2014.

[11] Federation of international robot-soccer association. http://www.fira.net/main/, 2014.

[12] Torneo mexicano de robótica. http://www.femexrobotica.org/tmr2014/, 2014.

[13] Manuel Hunter, Felipe Lara, Rafael Cisneros, and Juan Manuel Ibarra. Mechanical design and kinematic analysis of the ah1n1 humanoid robot. *Congreso Internacional en Electrónica, Comunicaciones y Computadoras (CONIELECOMP)*, 2011.

[14] Mark W. Spong and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, Inc., 1989.

[15] Donald Lee Pieper. *The Kinematics of Manipulators under Computer Control*. PhD thesis, Stanford University, 1968.

[16] Muhammad A. Ali, Andy Park, and C. S. George Lee. Closed-form inverse kinematic joint solution for humanoid robots. *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[17] Rafael Cisneros Limón. Estrategias de modelado cinemático y simulación en robots humanoides. Master's thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2009.

[18] Rafael Cisneros Limón. Robótica II. Universidad La Salle, 2011.

[19] Jacquelin Perry. *Gait Analysis: Normal and Pathological Function*. SLACK Incorporated, 1992.

[20] M. Vukobratovic, B. Borovac, and V. Potkonjak. Towards a unified understanding of basic notions and terms in humanoid robotics. *Robotica, 25, pp 87-101*, 2007.

[21] Héctor Montes Franceschi. *Análisis, diseño y evaluación de estrategias de control de fuerza en robots caminantes*. PhD thesis, Universidad Complutense de Madrid, 2006.

[22] Oscar Luis Vele G. Planificación de caminata para un robot bípedo. `http://loslocosrh.blogspot.mx/`, 2005.

[23] Sven Behnke. Online trajectory generation for omnidirectional biped walking. *The 2010 IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[24] Miomir Vukobratovic. Zero-moment point - thirty five years of its life. *International Journal of Humanoid Robotics, 1, pp 157-173*, 2004.

[25] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[26] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly Media, Inc., 2008.

[27] Martin Peris. Opencv: Stereo camera calibration. `http://blog.martinperis.com/2011/01/opencv-stereo-camera-calibration.html`, 2011.

[28] Hassan Munir. Calibrating stereo cameras. `http://cseautonomouscar2012.wordpress.com/2012/11/11/calibrating-stereo-cameras/`, 2012.

[29] Gustavo Alemán. Control de posición por visión artificial de brazo robótico de 4 gdl. Technical Report, Departamento de Control Automático del CINVESTAV-IPN, 2014.